Draft Minutes for April 29-May 3, 2019
MEETING OF ISO/IEC JTC 1/SC 22/WG 14 AND INCITS PL22.11
WG 14/N 2376

Dates and Times

29 April, 2019	09:00 – 12:00	Lunch	13:30 – 16:30
30 April, 2019	09:00 – 12:00	Lunch	13:30 – 16:30
1 May, 2019	09:00 – 12:00	Lunch	13:30 – 16:30
2 May, 2019	09:00 – 12:00	Lunch	13:30 – 16:30
3 May, 2019	08:30 – 11:00		

Meeting Location

BSI Group
Chiswick Tower
389 Chiswick High Road
London, W4 4AL
UK

Meeting information

Venue information: N 2308
Clive: Gave some information on fire exits, lunch, restrooms, etc.

Local contact information

Clive Pygott <clivepygott@gmail.com>

1. Opening Activities

1.1 Opening Comments (Pygott, Keaton)

Attendance sheets: 1 for the building, and for the committee.
Fire drill on Wednesday.

1.2 Introduction of Participants/Roll Call

Name	Organization	NB	Comments
David Keaton	Keaton Consulting	USA	WG14 Convener
Daniel Plakosh	SEI/CERT/CMU	USA	WG14 ISO eCommittee Secretary
Blaine Garst	Blastwave	USA	Phone
Rajan Bhakta	IBM	CA	PL22.11 Chair
Melanie Blower	Intel	USA	
Fred Tydeman	Tydeman Consulting	USA	PL22.11 Vice Chair
Tom Plum	Plum Hall	USA	Phone
Martin Sebor	Red Hat	USA	
Larry Jones	Siemens PLM Software	USA	WG 14 Project Editor; Phone
Aaron Ballman	GrammaTech	USA	
Clive Pygott	LDRA	UK	
Jens Gustedt	INRIA	France	WG14 Project Editor
Robert Seacord	NCC Group	USA	
Peter Sewell	U. Cambridge	UK	Memory model study group
David Svoboda	SEI	USA	Phone
Lars Bjonnes	Cisco	USA	
Niall Douglas	MayStreet	Ireland	Guest of WG21 Convener, Phone
Paul McKenney	IBM	USA	Phone
Maged Michael	IBM	USA	Phone
David Ward	MISRA	UK	MISRA Liaison; Phone
Nick Stoughton	USENIX	USA	Linux Liaison; Phone
Geoff Clare	USENIX	UK	POSIX Liaison; Phone
Philipp Krause	Albert-Ludwigs-Universität	Germany	
Martin Uecker	University of Gottingen	Germany	
Roberto Bagnara	University of Parma	Italy	
Derek Jones	Knowledge Software	UK	
Freek Wiedijk	Plum Hall	USA	
Alex Gilding	Perforce Software	USA	
Victor Gomes	U. Cambridge	UK	
Kayvan Memarian	U. Cambridge	UK	
Charles Wilson	Draeger Medical Systems	USA	Phone
Andrew Banks	LDRA	UK	

1.3 Procedures for this Meeting (Keaton)

The Meeting Chair and WG14 Convener, David Keaton, announced that procedures would be as per normal. Everyone was encouraged to participate in the discussion and straw polls.

Straw polls are an informal WG14 mechanism used to determine if there is consensus to pursue a technical approach or possibly drop a matter for lack of consensus. They are voted on by a show of hands for people that approve, reject or abstain, respectively (denoted by #approved/#reject/#abstain in the minutes) on the poll question. Straw polls are not formal votes, and do not in any way represent any National Body position. National Body positions are established in accordance with the procedures established by each National Body.

All 'N' document numbers in these minutes refer to JTC1 SC22/WG14 documents unless otherwise noted.

Participants need to register on ISO's site for this meeting.

David Keaton is the meeting Chair.
Rajan Bhakta is the Recording Secretary.

1.4 Approval of Previous Minutes [N 2307] (PL22.11 motion, WG 14 motion)

Amended by Fred's typo's corrections.
Motion (WG14, PL22.11): Bhakta/Tydeman
Passes by unanimous consent.

1.5 Review of Action Items and Resolutions

David: Request that ISO cancel the CPLEX project. – Done.
David: Appointing Aaron as chair of C Safety and Security Rules Study Group. Get ISO to give him an account for telecon purposes. – Done.
Aaron: The Safety and Security Study Group should propose by the next WG14 meeting whether it wants to drop safety or it has the legal issues handled. – Done.
Aaron: Need a permanent chair or a temporary chair for the Safety and Security Study Group by the next WG14 meeting. – Done.
David: Ask Martin U. If he wants to champion N1923. – Done.
David: Ask Thomas if he will champion trailing commas in macros (N2160). – Done.
Aaron: Add in a requirement to have underscore versions of the attributes to N2269. – Done.
Aaron: Update N2269 to make it normative that attributes have no semantic impact. – Done.
Aaron: Add in an example to show the string parameter in use for the deprecated attribute. – Done.
David: Tell Andrew what we need to do to move paper N2258 further. – Done.
David: Let Jacob know what the sentiment for N2285 is. – Done.
Rajan: Put in meeting minutes any items that will go into C2X from this meeting. – Done.
Rajan: Write up a proposal for precision control for NaN's. – Done (see Fred's paper).

1.6 Approval of Agenda [N 2370] (PL22.11 motion, WG 14 motion)

Request to add 8.1, Special Math Functions discussion to the agenda.

Motion (WG14, PL22.11): Approval of agenda as amended above. (Ballman/Tydeman)

Passes by unanimous consent.

1.7 Identify National Bodies Sending Experts

Canada, Germany, France, Italy, UK, USA

1.8 INCITS Antitrust Guidelines and Patent Policy

1.9 INCITS official designated member/alternate information

Can ask Rajan Bhakta to check if they are a member if needed by anyone.

2. Reports on Liaison Activities

2.1 SC 22

We have been overusing DR's and TC's from ISO's point of view. ISO considers a defect being something so severe it needs to be fixed in 6 weeks. We have been using it for a wider scope.

Now the standing document for procedures is only for standards, not for TC's as it was before.

What we used to handle as DR's before will now be handled as CR's (Clarification Requests)

2.2 PL22.11/WG 14

2.2.1 Document system

The document system has a lot of N documents from ISO's point of view. They normally only need it for procedural documents like ballot documents and minutes. We will be going to C<number> documents after this meeting.

For pre-Ithica, we'll have beta-testers for the C<number> documents. Post-Ithica we plan to have C<number> documents for all.

Martin Sebor has volunteered to update the web site.

2.3 PL22.16/WG 21

Continuing to move along with C++2A, continuing discussion on the mailing list.

Herb wants to say (paraphrased) the Direction Group is opposed to the direction for deterministic exceptions. He is working on a paper showing actual practice. Likely to be showed in the July meeting.

2.4 PL22

No responses for Chair yet.

2.5 WG 23

Three documents out for vote: main document (language independent), C, Ada: All passed and are in the process of being published. Starting relations with WG21 though SG12.

2.6 MISRA C

Defer to combine with MISRA discussions later.

2.7 Other Liaison Activities

3. Reports from Study Groups

3.1 C Floating Point activity report

- Working closely with 754 (strong liaison, one member is chair and one is the editor)
- Working on creating updated TS's for the ones voted in (parts 2 and 3) for inclusion into the working C2X draft
  - Planning on making part 2 _Decimal* types optional like _Complex
- Will continue binding IEEE-754 2019 to the TS's that remain after C2X

Other comments: Mike from the C FP group: The preferred quantum exponents table in part 2 was really nice.

3.2 C Safety and Security Rules Study Group

Permanent chair is Charles Wilson.

We will work out the licensing issue.

We will discuss some of this on Tuesday as we're going through the MISRA C rules (slow progress).

3.3 C Memory Object Model Study Group

Quite active in the last 6 months. For provenance, we have reasonable detail.

Discussion with C++ and others too.

4. Teleconference Meeting Reports

4.1 Report on any teleconference meetings held

None.

5. Future Meetings

5.1 Future Meeting Schedule

- 21-25 October, 2019 – Ithaca, New York, US [N 2327]
- Spring, 2020 – Freiburg, Germany (tentative)
  - Philip: A 550-year-old university, warmest part of Germany. Easy to get to via train. Closest airport is Frankfurt.
- Fall, 2020 – TBD (North America)
  - Peter: Some issues with meetings inside USA due to border issues.
  - David: Yes, some people can't get in USA others can't leave so we try to distribute the "pain".
- Spring, 2021 – Strasbourg, France (tentative)
  - Jens: Similar to Freiburg. Host of the European parliament and court of human rights. Train/bus/flight connections to the Paris, Frankfurt and Amsterdam airports. All major airline alliances have a bus/train as an integral part of the ticket.

5.2 Future Mailings

- Post-London – 3 June 2019
- Pre-Ithaca – 23 September 2019
- Post-Ithaca – 18 November 2019

6. Document Review

6.1 Ballman, Attributes in C (updating N2269) [N 2335]

Result: Consensus to add to C2x.

6.2 Ballman, Propose adopting additional previously-discussed attribute papers [N 2267] [N 2268] [N 2270]

N2267: nodiscard

Result: Consensus to add to C2x.

N2268: fallthrough

Result: No consensus to add to C2x.

N2270: (maybe_unused):

Result: Consensus to add to C2x.

6.3 Ballman, Querying attribute support [N 2333]

Result: Aaron to continue with this approach.

Slightly in favor of having __has_c_attribute be a function-like macro.

6.4 Ballman, The deprecated attribute (updating N2266) [N 2334]

Result: Consensus to add to C2x.

6.5 Editors' discussions

Gustedt, ISO/IEC 9899 editor report March 2019 [N 2348]

Gustedt, ISO/IEC 9899 working draft March 2019, diffmarks [N 2347]

Gustedt, ISO/IEC 9899 working draft March 2019 [N 2346]

Part 2 of the floating-point TS's should be in the post meeting mailing.

Gustedt, Remove conditional "WANT" macros from numbered clauses [N 2359]

Result: Feature test macros for each of the library clause sections as described in N2359.

6.6 Bhakta, The overloading of semi-colons in the C standard specification [N 2345]

OK with the change.

Leaving it to the editor to clearly change this in a way that make it clear.

6.7 Svoboda, Defining Undefined Behavior [N 2365]

General discussion. No consensus on any movement on this.

6.8 Seacord, Bounds-checking Interfaces: Field Experience and Future Directions [N 2336]

General disagreement on how to proceed. Strong even split on keeping Annex K and on removing it.

6.9 Douglas, Enhanced C/C++ memory and object model [N 2367]

General discussion and consensus on the need to rework this.

6.10 C Memory Object Model Study Group discussions

Sewell, Exploring C Semantics and Pointer Provenance [N 2311]

Sewell, Moving to a provenance-aware memory model for C: proposal for C2x [N 2362]

Sewell, C provenance semantics: examples [N 2363]:

Sewell, C provenance semantics: detailed semantics [N 2364]:

Long and spirited discussion.

Result: General consensus on agreement on the direction of the provenance proposal.

6.11 Gustedt, Introduce the term storage instance [N 2328]

Result: General consensus on agreement on the direction of this proposal.

6.12 MISRA Discussions

Result: Action items:

Charles to ask the C Safety and Security study group what they want to do under the assumption of not being able to use MISRA.

David K to look into mechanisms for publishing the MISRA documents under ISO and discuss with Andrew Banks.

Banks, Follow-up on enumerating & cross referencing Annex J (formerly [N 2112])

Result: Consensus to uniquely label the bullet points in Annex J.

Banks, Follow-up on "defensive" attribute (formerly [N 2258])

Result: No consensus to move along these lines.

6.13 Stoughton, Proposal To Add Extended Month Name Formats to strftime() [N 2337]

Result: Incorporate N2337 into C2X switching the cases for the b/B's.

6.14 Stoughton, Error Indicator For Encoding Errors In fgetwc [N 2338]

Result: Incorporate N2338 into C2X.

6.15 Stoughton, Change Request for fopen exclusive access [N 2357]

Result: No consensus (but no significant opposition) to the second change.

No objections to making a change to 7.21.5.3 p5.

6.16 Integrating floating-point TS updates into C2x

Thomas, TS 18661-1 plus CR/DRs for C2X [N 2314]
Thomas, TS 18661-1 plus CR/DRs for C2X with change marks [N 2315]
Thomas, TS 18661-2 plus CR/DRs for C2X [N 2341]
Thomas, TS 18661-3 as annex [N 2342]
N2374 (slide deck)

Thomas, C2X proposal - TS 18661-4a [N 2355]

N2373 (slide deck)
Result: Adopt 18661 part 4a (no reduction functions) into C2X.

Thomas, update for C2X payload functions [N 2356]

Result: Adopt N2356 into C2X.

6.17 Tydeman, FE_TONEARESTFROMZERO w.r.t. FLT_ROUNDS [N 2319]

Result: N2319 to be put into C2X.

6.18 Tydeman, Precision and NAN/NAN(...) and INF/INFINITY [N 2320]

Result: Want something along the lines of the macro addition part of N2320 to be put into C2X. Requesting a new paper for better wording for this.

6.19 Tydeman, Nextafter/nexttoward/nextup/nextdown [N 2321]

The chairman for IEEE has a use case for the existing words for this so withdrawing this change request.

6.20 Tydeman, SD3 9: PreProcessor unspecified line numbers [N 2322]

Result: N2322 to be put into C2X.

6.21 Tydeman, SD3 1: DR 440: Test macros for FP being 754 types [N 2323]

Result: Want something like N2323 option 1 to be put into C2X.

6.22 Tydeman, SD3 13: DR 482: Macro span files: undefined [N 2324]

Result: Fred to follow up on this. New document N2379.

6.23 Tydeman, SD3 11: Maximum normalized FP number [N 2325]

Result: N2325 be put into C2X.

6.24 Tydeman, Merge DR 432+467 [N 2326]

Result: N2326 be put into C2X with editorial changes on the footnote expected.

6.25 Gustedt, intmax_t, a way out [N 2303]

Result: Clear direction to have extended integer types that are wider than [u]intmax_t.

6.26 Gustedt, Clean up atomics [N 2329]

Result: Happy with the proposed non-normative changes (section 2.1) in N2329.

No consensus on adding into C2X the proposed additions for the synchronization of the library functions proposed in section 2.2.1 of N2329.

No full consensus on using the atomic_fetch_OP_explicit functions as a model definition for all the other compound assignment operations as proposed.

No consensus to introduce the new atomic_OP_fetch[_explicit] operations as complement for all OP= operators and adding ordering as specified in section 2.2.3 of N2329 into C2X.

No consensus to add in the selection rules for atomic generic functions as given in section 2.2.5 of N2329 into C2X.

6.27 Gustedt, Moving to two's complement sign representation [N 2330]

Result: Direction to fix the minimum value of signed integer types to the negated power of 2 of the width – 1.

No consensus on the proposed changes for bit-fields.

Agree to the general direction proposed for specifying the bounds of integer types based on widths as specified in N2330.

6.28 Gustedt, Unify string representation functions [N 2360]

Result: Prefer a type generic interface to print all basic types into a character buffer as specified in N2360.

No consensus to restrict the format of the tostr functions to string literals?

Consensus to allow a default argument for the format method of overloading for the tostr functions.

6.29 Gustedt, Out-of-band bit for exceptional return and errno replacement [N 2361]

No time to discuss.

6.30 Gustedt, Align spelling of keywords with C++ and make them feature tests [N 2368]

Result: Want to go in the general direction proposed by N2368.

The following list of keyword and macros should be subject to N2368:

alignas, alignof, bool, static_assert, thread_local, false, true

6.31 Sebor, Toward more efficient string copying and concatenation [N 2349]
Result: Put N2349 into C2X.

6.32 Sebor, Defining new types in offsetof [N 2350]
Result: Put N2350 into C2X.

6.33 Sebor, Add strnlen to C2X [N 2351]
Result: No consensus on putting N2351 into C2X.

6.34 Sebor, Add stpcpy, and stpncpy to C2X [N 2352]
Result: No consensus to put N2352 into C2X.

6.35 Sebor, Add strdup and strndup to C2X [N 2353]
Result: N2353 be put into C2X.
The committee wants a proposal for the wide character versions of any POSIX functions voted in this meeting.

6.36 Sebor, Constraints on parameters to main [N 2354]
Result: No consensus to put N2354 into C2X.
No consensus to ban calling main recursively for hosted implementations.
It is intended that the constraints in 5.1.2.1.1#2 apply only on the initial call to main.

6.37 McKenney, Pointer lifetime-end zap [N 2369]
Result: Preference for possible resolutions among the ones proposed in N2369:
1: 13
2: 11
3: 13
4: 8
5: 0

6.38 Krause, No internal state for mblen [N 2358]
Result: Put N2358 in C2X.

6.39 Uecker, Improved Rules for Tag Compatibility [N 2366]
Revision of N2105.
Result: Want to see more papers like N2366.

7. Clarification Requests

7.1 Discussion on the Clarification Request Process

Long discussion on what to do.
General consensus to continue handling clarification requests.

7.2 IS 9899:2011/9899:2018 Clarification Requests [N 2316]

Action Item: Blaine: Ensure all DR's marked C2X are in the latest working draft (N2346).
DR501: Making DECIMAL_DIG obsolescent
Result: Move DR501 to C2X.
DR496: offsetof questions
Result: Move DR496 to C2X.

7.3 TS 18661 Clarification Requests [N 2317]

Result:
DR13: Move to C2X (C FP action item).
DR16: Move to C2X (C FP action item).
DR20-25: Move to C2X (C FP action item).

Thomas, P4 CR for rootn case differs from IEEE 754 [N 2309]

Result: Assigned to DR26.
Put into TS 18661-4 and C2X (C FP action item).

8. Other Business

8.1 Special math functions (ISO/IEC 24747)

Result: The __STDC_MATH_SPEC_FUNCS__ macro is useful as is. Consensus to keep it as is.

9. Resolutions and Decisions reached

9.1 Review of Decisions Reached

Straw poll: Should N2335 be put into C2x Working Draft? Yes.

Straw poll: Should N2267 be put into the C2x Working Draft? Yes.

Straw poll: Should N2268 be put into the C2x Working Draft? No consensus.

Straw poll: Should N2270 be put into the C2x Working Draft? Yes.

Straw poll: Is the committee in favor of something along the lines of __has_c_attribute as given in N2333? Weak agreement.

Straw poll: Is the committee in favor of having __has_c_attribute be a function-like macro? Yes.

Straw poll: Should N2334 be put into the C2x Working Draft? Yes.

Straw poll: Should we remove the WANT macros from the numbered clauses of the standard (keep it in Annex F)? Yes.

Straw poll: Does the committee want feature test macros for each of the library clause sections as described in N2359? Yes.

Straw poll: Does the committee want to remove or deprecate Annex K in C2X? No consensus.

Straw poll: Is the committee OK with the general direction of the provenance proposal? Yes.

Straw poll: Does the working group agree to uniquely label the bullet points in Annex J? Yes.

Straw poll: Do we want to see something along the lines of N2258? No consensus.

Straw poll: Do we want to incorporate N2337 into C2X switching the cases for the b/B's? Yes.

Straw poll: Do we want to incorporate N2338 into C2X? Yes

Straw poll: Do we want to incorporate N2357's second proposed change into C2X? No consensus.

Straw poll: Do we want any changes to 7.21.5.3 p5 regarding the "exclusive" in exclusive file access? No objections to making changes.

Straw poll: Should we adopt 18661 part 4a (not reduction functions) into C2X (not consider naming issue)? Yes.

Straw poll: Should we adopt N2356 into C2X? Yes.

Straw poll: Do we want N2319 to be put into C2X? Yes.

Straw poll: Do we want something along the lines of the macro addition part of N2320 to be put into C2X? Yes, we want a new paper with better wording for this.

Straw poll: Do we want N2322 to be put into C2X? Yes.

Straw poll: Do we want something like N2323 to be put into C2X? Yes.

Straw poll: Do we want something like to option 1 or option 2 in N2323 to be put into C2X? We want something like option 1.

Straw poll: Should N2325 be put into C2X? Yes.

Straw poll: Should N2326 be put into C2X with editorial changes on the footnote expected? Yes.

Straw poll: Does WG14 agree to have extended integer types that are wider than [u]intmax_t? Yes.

Straw poll: Is WG14 happy with the proposed non-normative changes (section 2.1) in N2329? Yes.

Straw poll: Is WG14 happy adding into C2X the proposed additions for the synchronization of the library functions proposed in section 2.2.1 of N2329? No consensus.

Straw poll: Shall we use the atomic_fetch_OP_explicit functions as a model definition for all the other compound assignment operations as proposed in N2329? Slight preference, Jens to come back with this.

Straw poll: Should we introduce the new atomic_OP_fetch[_explicit] operations as complement for all OP= operators and adding ordering as specified in section 2.2.3 of N2329 into C2X? No consensus.

Straw poll: Do we want to add in the selection rules for atomic generic functions as given in section 2.2.5 of N2329 into C2X? No consensus.

Straw poll: Does WG14 agree to fix the minimum value of signed integer types to the negated power of 2 of the width - 1? Yes.

Straw poll: Does WG14 agree to proposed changes for bit-fields as specified in N2330? No consensus.

Straw poll: Does WG14 agree to the general direction proposed for specifying the bounds of integer types based on widths as specified in N2330? Yes.

Straw poll: Does the committee want to go in the general direction proposed by N2368? Yes.

Straw poll: Does WG14 agree to the following list of keyword and macros as being subject to N2368:

alignas, alignof, bool, static_assert, thread_local, false, true

Yes.

Straw poll: Do we want to put N2349 into C2X? Yes.

Straw poll: Should N2351 be put into C2X? No consensus.

Straw poll: Should N2352 be put into C2X? No consensus.

Straw poll: Should N2353 be put into C2X? Yes.

Straw poll: Does the committee want a proposal for the wide character versions of any POSIX functions voted in this meeting? Yes.

Straw poll: Do we want to put N2354 into C2X? No consensus.

Straw poll: Does the committee want to ban calling main recursively for hosted implementations? No consensus.

Straw poll: Is it intended that the constraints in 5.1.2.1.1#2 apply only on the initial call to main? Yes.

Straw poll: Which possible resolution does the committee want among the ones proposed in N2369?

1: 13
2: 11
3: 13
4: 8
5: 0

No clear consensus on one resolution.

Straw poll: Does the committee want to see a future paper along the lines of N2366? Yes.

Straw poll: Should N2309 be put into TS 18661-4 and C2X? Yes.

Straw poll: Do we want N2358 in C2X? Yes.

Straw poll: Should N2350 be put into C2X? Yes.

Straw poll: __STDC_MATH_SPEC_FUNCS__ macro is useful as is. Does the committee want to keep it as is? Yes. Straw poll: Does WG14 like a type generic interface to print all basic types into a character buffer as specified in N2360? Yes.

Straw poll: Does WG14 want to restrict the format of the tostr functions to string literals? No consensus.

Straw poll: Does WG14 like the default argument format method of overloading for the tostr functions? Yes.

Straw poll: Is WG14 good with the general direction of N2328? Yes.

9.2 Review of Action Items

Carry over:

None.

New:

Charles: Ask the C Safety and Security study group what they want to do under the assumption of not being able to use MISRA.

David K: Look into mechanisms for publishing the MISRA documents under ISO and discuss with Andrew Banks.

Blaine: Ensure all DR's marked C2X are in the latest working draft (N2346).

Editor: Move DR501 to C2X.

Editor: Move DR496 to C2X.

Editor: Put N2335 into C2X.

Editor: Put N2267 into the C2X.

Editor: Put N2270 into the C2X.

Editor: Put N2334 into the C2X.

Editor: Uniquely label the bullet points in Annex J.

Editor: Put N2337 into C2X switching the cases for the b/B's.

Editor: Put N2338 into C2X.

C FP: Give 18661 part 4a (not reduction functions) for inclusion into C2X.

Editor: Put N2356 into C2X.

Editor: Put N2319 into C2X.

Editor: Put N2322 into C2X.

Editor: Put N2325 into C2X.

Editor: Put N2326 into C2X with editorial changes on the footnote expected.

Editor: Put N2349 into C2X.

Editor: Put N2353 into C2X.

C FP: Put N2309 into TS 18661-4 and C2X.

Editor: Put N2358 into C2X.

Editor: Put N2350 into C2X.

9.3 Identification of PL22.11 Voting Members

Cisco: Lars Bjonnes
GrammaTech: Aaron Ballman
IBM: Rajan Bhakta (Primary)
IBM: Paul McKenney
IBM: Maged Michael
Intel: Melanie Blower
Keaton Consulting: David Keaton
LDRA: Clive Pygott
Plum Hall: Tom Plum (Primary)
Plum Hall: Freek Wiedijk
Red Hat: Martin Sebor
SEI/CERT/CMU: Daniel Plakosh
SEI/CERT/CMU: David Svoboda
Tydeman Consulting: Fred Tydeman

9.3.1 Members Attaining initial Voting Rights at this Meeting

None

9.3.2 Members who regained voting rights

None

9.4 PL22.11 Voting Members in Jeopardy

9.4.1 Members in jeopardy due to failure to vote on Letter Ballots

None

9.4.2 Members in jeopardy due to failure to attend Meetings

None

9.4.2.1 Members (in jeopardy) who retained voting rights by attending this meeting

None

9.4.2.2 Members (in jeopardy) who lost voting rights for failure to attend this meeting

None

9.5 PL22.11 Non-voting Members

9.5.1 Prospective PL22.11 Members Attending their First Meeting

Blastwave: Blaine Garst

Perforce Software/Programming Research Ltd: Alex Gilding

9.5.2 Advisory members who are attending this meeting

Draeger Medical Systems: Charles Wilson

10. Thanks to Host

11. Adjournment (PL22.11 motion)

(Ballman/Tydeman)

No objections.

Adjourned at 10:34 am, Friday May 3rd, 2019

Appendix A: Details on Discussion

2. Reports on Liaison Activities

2.1 SC 22

David: We have been overusing DR's and TC's from ISO's point of view. ISO considers a defect being something so severe it needs to be fixed in 6 weeks. We have been using it for a wider scope.

Now the standing document for procedures is only for standards, not for TC's as it was before.

Derek: Are we backing off on DR's?

David: No, we're doing it under CR's now (Clarification Requests)

2.2 PL22.11/WG 14

2.2.1 Document system

David: The document system has a lot of N documents from ISO's point of view. They normally only need it for procedural documents like ballot documents and minutes. We will be going to C# documents after this meeting.

Niall: Can we use what WG21 is doing (Howell's automated system for P documents).

David: If we can we will.

For pre-Ithica, we'll have beta-testers for the C# documents. Post-Ithica we plan to have C# documents for all.

Aaron: Is there anything that prevents us from publishing C# documents even if it is the same as an N# document other than the document name/number?

David: No, we can't have it publically available.

Aaron: Do we need to do anything special for the GIT repo?

Jens: It is password protected so it is not public.

David: Had to go to Geneva to object to their removal of their Oxford comma from our work. They allowed us to do whatever we do as long as it is done consistently.

Jens: We are in need of a good webpage.

David: Martin Sebor has volunteered to update it. I have asked Keld to give him access to change things.

Martin S: OK with others to funnel through me.

2.3 PL22.16/WG 21

Aaron: Continuing to move along with C++2A, continuing discussion on the mailing list.

Niall: Herb wants to say (paraphrased) the Direction Group is opposed to the direction for deterministic exceptions. He is working on a paper showing actual practice. Likely to be showed in the July meeting.

2.4 PL22

No responses for Chair yet.

2.5 WG 23

Clive: 3 documents out for vote: main document (language independent), C, Ada: All passed and are in the process of being published. Starting relations with WG21 though SG12.

2.6 MISRA C (defer to combine with MISRA discussions later)

2.7 Other Liaison Activities

3. Reports from Study Groups

3.1 C Floating Point activity report

- Working closely with 754 (strong liaison, one member is chair and one is the editor)
- Working on creating updated TS's for the ones voted in (parts 2 and 3) for inclusion into the working C2X draft
  - Planning on making part 2 _Decimal* types optional like _Complex
- Will continue binding IEEE-754 2019 to the TS's that remain after C2X

Comments: Mike: The preferred quantum exponents table in part 2 was really nice.

3.2 C Safety and Security Rules Study Group

Clive: We are a SC22 study group, not a WG14 study group according our chair so problems with getting a Zoom account.

David: No, wrong contact at ISO.

Clive: Permanent chair is Charles Wilson.

We will work out the licensing issue.

We will discuss some of this on Tuesday as we're going through the MISRA C rules (slow progress).

We either get all safety or a few security, not both. Not many calling in to the meetings.

David: Hopefully we can resolve this tomorrow.

Roberto: There is also the problem with the fact that MISRA rules have been considered one at a time, and behind each rule there are years of work. The discussions really depend on who is on the call and we should make decisions based on technical merits and we should not depend on the members that ended up calling in.

David: The charter of the study group is to add safety into the document (security is already there). It doesn't necessarily need MISRA since it can be from any source so they need to be there to avoid removing any.

Roberto: The process is a key part of the rules; they cannot be taken in isolation.

David: The goal is not to add in MISRA, it is to add in safety.

Clive: Two different views of safety. One is technical, like type safety and analysis of code. The other view is bringing in the softer aspects like experience and mistakes

programmers bring in. One view is what can we bring into the compiler to warn/catch these. The other issue is the code reviewable, etc. which are not things the compiler would be warning about. The differences between these two are likely a cause of a lot of disagreement.

Martin S: Those softer aspects were explicitly not part of the document.

Clive: That brings up issues of a hostile reading of the document. I have issues with that.

Derek: The original MISRA had a mailing list which I may have archived. A lot of issues are the group has an ignorance of C.

3.3 C Memory Object Model Study Group

Peter: Quite active in the last 6 months. For provenance, we have reasonable detail. Discussion with C++ and others too. There are slides which will be passed around during the discussion time for this.

4. Teleconference Meeting Reports

4.1 Report on any teleconference meetings held

None.

5. Future Meetings

5.1 Future Meeting Schedule

- 21-25 October, 2019 – Ithaca, New York, US [N 2327]
- Spring, 2020 – Freiburg, Germany (tentative)
  - Philip: A 550-year-old university, warmest part of Germany. Easy to get to via train. Closest airport is Frankfurt.
- Fall, 2020 – TBD (North America)
  - Peter: Some issues with meetings inside USA due to border issues.
  - David: Yes, some people can't get in USA others can't leave so we try to distribute the "pain".
- Spring, 2021 – Strasbourg, France (tentative)
  - Jens: Similar to Frieburg. Host of the European parliament and court of human rights. Connected to the Paris airport. Luftansa has a bus/train as an integral part of the ticket.

5.2 Future Mailings

- Post-London – 3 June 2019
- Pre-Ithaca – 23 September 2019
- Post-Ithaca – 18 November 2019

6. Document Review

Monday morning

6.1 Ballman, Attributes in C (updating N2269) [N 2335]

Rajan: Minor: Paragraph numbers in changelog are wrong (p4 doesn't talk about semantic impact and p5 should be p4)

Aaron: Only two normative changes. (__ spelling and 'no semantic impact')

No semantic impact is 6.7.11p2

Philipp: Why have __ version if not required?

Aaron: For library implementers to allow working with user macros.

Jens: GCC does this for their attributes.

Melanie: Is it required to add the vendor prefix?

Aaron: Only for vendor attributes.

Martin S: Does the __ prefix apply to the namespaces?

Aaron: It is up to the vendors for their attributes. Not specified.

*Straw poll: Should N2335 be put into C2x Working Draft?

20/0/1 (In favor, Opposed, Abstain)

Result: Consensus to add to C2x.

6.2 Ballman, Propose adopting additional previously-discussed attribute papers [N 2267] [N 2268] [N 2270]

N2267: nodiscard

Freek: Why is this a programming mistake (malloc)?

Aaron: That's why 'malloc' was an example, like for empty in C++.

Martin S: Is anyone planning on applying this to the C Standard library?

Aaron: I am interested, but not sure I can.

*Straw poll: Should N2267 be put into the C2x Working Draft?

21/0/1 (In favor, Opposed, Abstain)

Result: Consensus to add to C2x.

N2268: fallthrough

Philip: Don't like the idea of having something that will add warnings to existing code (code without the attribute will now have a diagnostic).

Aaron: The compilers that implement support for this do not do it by default, it is an opt in.

Philipp: Once it is in the standard it will be on by default.

Aaron: Everyone seems to find a lot of false positives, but they all seem to find one valid problem.

Clive: If the language doesn't have this we or MISRA would likely add something like this, but probably the opposite (add in nofallthrough).

Freek: What about cases where you intend to fallthrough obviously (stacked cases)?

Aaron: Yes, it is in the example in the paper.

Jens: If there was an empty statement (;)?

Aaron: I would expect a diagnostic.

Martin S: The one use case that got a lot of bug reports was for finite state machines. Users asked for an attribute for the entire switch statement. Do you know if that's implemented?

Aaron: No, not that I'm aware of.

Derek: The constraint violation is macro unfriendly (see example in the paper).

Aaron: Can a macro expand to a case label?

Derek/Jens: Yes.

Aaron: This is matching C++.

Derek: But C++ is anti-macro.

Aaron: So am I!

Martin S: Why is it a constraint violation?

Aaron: Because there is nothing following. You can't fallthrough to nothing.

Jens: Why is that bad?

Aaron: This is to show programmer intent.

Robert S: Can this be anywhere?

Aaron: No, it has to be before a case label in the same switch statement.

Niall: In C++ for macro expansions that could be null, the idea was to add empty semi-colons after the macro invocation.

Derek: Generally compilers warn on empty statements.

Niall: A common extension was to have a fallthrough comment that had the same semantics as this.

Martin S: Yes, GCC has a regex for this.

Jens: If we vote this in, there is no implementation that is conforming.

Aaron: Yes, with recommended practice, both Clang and GCC don't diagnose the last case in the example.

Roberto: Is it a constraint violation to apply no-discard on a void function?

Aaron: I think it should. Actually, no, we don't have it under any recommended practice.

Aaron: In this case, I am different from C++ commonality since it is unlikely to be common code between C and C++.

Derek: What happens for macro expanding to something like an 'if' statement?

```
case 1:
  ..
  ..
  H()
    case 2: x = 3;
```

Aaron: Not sure, but not an interesting case for my purposes.

Freek: Constraint violation means stop compiling?

Jens: No. Just has to diagnose.

Derek: Generally means stop compile.

Lars: C++ would stop compilation.

Aaron: Yes, C doesn't have an equivalent form.

Robert S: The fact that this is new means this should not affect existing code.

Aaron: Yes.

Martin S: I expect existing implementations would hide this under a macro.

Roberto: If you have a constraint violation, we don't have a precise definition about it.

Aaron: I believe we do.

Freek: Is a fallthrough before the first case label allowed?

Aaron: Yes.

Jens: We should put it on the case label. I understand your constraints were to match C++.

Aaron: This was based on GNU in C++ which didn't do it on the case label I think.

Martin S: I think the C++ attribute predates GNU.

Philipp/Jens: There is fallthrough that doesn't have syntax with this attribute.

Aaron: Do not take a straw poll and let me look into the constraint violation.

Does anyone want to diverge from C++ and have this apply to the case statement.

Philipp/Jens: I don't want the attribute at all.

David: This committee has rightly had a bias to match C++. If we add a feature that C++ has, we should match them.

Clive: Can someone go back to C++ to bring up what we said and see if they are willing to change?

Aaron: Unlikely to happen.

Jens: If we don't have this in C, it can be an extension for C compilers if they want.

David: But the compiler gets to choose, not the user.

*Straw poll: Should N2268 be put into the C2x Working Draft?

14/3/6 (In favor, Opposed, Abstain)

David: What do you think your national body will vote for this?

Canada: Rajan: No opinion at this time.

Germany: Philipp: Not important enough to be against the whole standard.

France: Jens: Not important enough to be against the whole standard.

Italy: Roberto: No problems.

UK: Derek/Peter/Clive: No problems.

USA: Looks like majority in favor.

Based on the country information, it seems there is no consensus.

David: We did have some new issues this meeting. I would like to see them addressed. This means we will discuss it at most one more meeting.

Result: No consensus to add to C2x.

N2270: (maybe_unused):

Jens: Does this apply to function parameters?

Aaron: Yes.

Freek: This is a C++ compatibility thing?

Aaron: Yes.

Robert S: What happens if you have a comma separated list?

Aaron: Depends on where you put the attribute.

Roberto: If it was int [[maybe_unused]] i, j, k;?

Aaron: Ill-formed. As with blocks (Freek).

Martin S: Since it can only be there once per attribute list doesn't that cause a problem?

Aaron: This is to disallow the list form of attributes. It is a general attribute thing.

Martin S: GCC attributes don't have this restriction. What's our general view on these constraints violations on attributes?

Aaron: Clang does diagnose this. They should be constraint violations like other constraint violations in the standard.

Roberto: The nodiscard on a void function is not a constraint violation, but this one is. It's inconsistent.

Aaron: They are different levels of constraint violations. The first is a logic error that I will probably write a paper to make it a violation for C and C++, while this is for the attribute syntax.

Martin S: I am uneasy about adding a hard constraint for a feature that is mainly for diagnostics.

Aaron: I personally agree, but it is matching what C++ does. C++ shouldn't disallow multiple attributes in the same list.

Martin S: Can we give you an action item to fix their standard?

Aaron: I don't think they would do it and I don't want to go in front of the EWG to do so.

Martin S: There are a number of issues with attributes that need to be resolved in C++.

Jens: Maybe not make it a constraint violation but make it recommended practice.

Aaron: Would be different from C++.

Freek: Can make it an obligatory warning, not a constraint violation even though it really is the same thing.

Aaron: That is what this is intended.

Jens: We define the term diagnostic. We can say we want a diagnostic and not imply you have to stop translating.

*Straw poll: Should N2270 be put into the C2x Working Draft?

21/0/2 (In favor, Opposed, Abstain)

David: Any objections to making lunch 1h to go over the next two papers?

None. Break until 1:00 pm.

6.3 Ballman, Querying attribute support [N 2333]

Aaron: Since vendors can have various attributes, this is a mechanism to allow programmers to query if it is supported. Very similar to 'defined'. Current paper is familiarity to C++.

Martin: I don't understand the rational to have different preprocessor keywords for C and C++. You need 3 tests instead of 1.

Aaron: In practical terms, we already have __has_attribute so we can't use that. Also, C++ has standardized on the __has_cpp_attribute.

Niall: cpp explicitly stands for C Pre-Processor, not C++ so it can be used.

Martin: It is a mistake in C++ and it can be fixed (add a new one and keep the existing one).

Jens: This is a special case; it is a macro and a preprocessor keyword. Why was it not a function like macro?

Niall: It is a result of a long torturous process that no one wants to go through again. Compilers had already implemented it before it was fully standardized and it became a defacto standard and so it is the way it is.

Aaron: I am OK if it was changed so it became a function-like macro.

Blaine: There is a broader question about finding out what the compiler chose at runtime or compile time. There is no good way to find that out. I thought C++ had something for this too.

Aaron: There was some discussion and they added __has_include, but nothing in the standard and they backed off of this.

Aaron: On the reflector, there was pushback on having this at all and instead use STDC_VERSION. Do we want something like this or something completely different? Also, how much concern is there for this behaving like 'defined'

Jens: I like the idea of having something like this as every compiler will add in attributes. I would also be in favor of having something more general too. I don't like the syntax as defined, I would like it to be a macro.

Roberto: My experience with the STDC macro is that it is inaccurate.

Blaine: I would rather see a single header file with all the macro's. The list of things to ask about should be in the header file.

Aaron: An implementation concern is you need multiple header files or macros.

Blaine: The syntax change is worse.

Aaron: What about vendor attributes?

Blaine: They have other options. For C Standard attributes, they should not be changing.

Aaron: My point was command line arguments/flags can change whether __has_attribute returns 0 or something else for example.

Martin: How this integrates with third party attributes and tools is what I'm worried about.

Aaron: There is work to be done either way.

Jens: I don't like mixing language and library due to sync issues.

Philipp: Seems to work for Linux like linux.h

Aaron: Right now, they only need to know how to spell the attribute, for macros they need a much higher cognitive load.

Blaine: I see a difference between C standard attributes and others. Vendor attributes are free to do whatever they want to do like extending what was used for C standard attributes.

Aaron: The typical workflow is vendors add an attribute in their namespace and then then when it is standardized, each vendor would have to change how they query their support to fit the newly standardized one.

Blaine: I disagree, we have different hypothesis for this.

*Straw poll: Is the committee in favor of something along the lines of __has_c_attribute as given in N2333?

16/5/4 (In favor, Opposed, Abstain)

Slight consensus. Aaron OK with continuing with this approach.

*Straw poll: Is the committee in favor of having __has_c_attribute be a function-like macro?

8/1/lots

Niall: Ask Richard Smith why it is the way it is.

Blaine: This bothers me. The C committee should be focusing on standardizing existing practice not designing by committee.

Jens: Having a list of what implementations do would be useful.

Roberto: Do we have other compiler vendors here?

Rajan: I took the result as a slight preference for function like macros, and most people (the “lots”) not caring either way.

6.4 Ballman, The deprecated attribute (updating N2266) [N 2334]

Clive: Other than triggering a diagnostic, is there any link between the standard deprecated features and this paper.

Jens: We don’t have any deprecated/obsolescent interfaces in the standard anymore.

Philipp: ATOMIC_VAR_INIT is a macro otherwise it would be one.

Aaron: So far no attributes has a meaning for duplicated but someday maybe we will.

*Straw poll: Should N2334 be put into the C2x Working Draft?

22/0/1 (In favor, Opposed, Abstain)

Monday afternoon

6.5 Editors' discussions

Gustedt, ISO/IEC 9899 editor report March 2019 [N 2348]

Section 2.2: Seems OK.

Section 3.1: Blaine: Can you put “must” “shall” and things like that in some special form like italics.

Undefined behavior: David: This is a much larger issue than editorial. We can’t do this for the existing standard since there is a lot of implicit undefined behavior.

Aaron: This is more for future changes.

Roberto: Ex. Shifting something too much is not high verification complexity but should stay undefined behavior.

Derek: If you are looking for a list of implicit undefined behavior I have submitted a list in the past.

Section 3.2: Blaine: I had trouble seeing my LaTeX changes shown in the C standard.

Jens: For small changes, non-LaTeX changes is fine, for large things LaTeX is appreciated. Ex. Attributes in LaTeX vs the individual attributes.

Derek: The ISO rules for ‘shall’ are different to how C uses it.

Gustedt, ISO/IEC 9899 working draft March 2019, diffmarks [N 2347]

Gustedt, ISO/IEC 9899 working draft March 2019 [N 2346]

Jens: Part 2 of the floating-point TS’s should be in the post meeting mailing.

Gustedt, Remove conditional "WANT" macros from numbered clauses [N 2359]

Rajan:

1) Remove WANT macros: OK

2) Header version: Seems fine.

3) Renaming functions: Against, since already implemented as is, names fit with pre-part 1 C, consistent with existing C standard, names fit function

For the 'from' functions, we didn't do 'to' for consistency. Ex. strftime

Fromfp takes three arguments, it is not a simple conversion. It can't be generalized due to the number of arguments (3 for fromfp due to rounding mode and width arguments).

4) Reserving prefixes: Seems fine.

Section 2: The C committee asked for the WANT macros. Now it makes no sense.

*Straw poll: Should we remove the WANT macros from the numbered clauses of the standard (keep it in Annex F)?

No objections.

Philipp: Prefer having a separate header for a large number of new arguably special purpose functions.

Section 3: Philipp: Prefer having _H in the macro to indicate it is a header.

Aaron: Is a header sufficient resolution for this? WG21 does this through classes of functionality.

Rajan: Assuming the no #undef properties holds too.

Aaron: Is per header the right granularity or should it be stdc_library_version?

Jens: I think the current proposal is the right level of granularity.

Martin: I am not too excited to add in these macros just as an interim measure until everyone supports the standard the functionality was added into.

Jens: We will have C3X and future standards so it is useful there too.

Philipp: It is not just a short transition period necessarily. Some compilers like for embedded systems may not have full support until much later but have partial support earlier. For free standing implementations, they can do parts as well.

Aaron: Clang doesn't ship a standard library so this would allow us to define STDC_VERSION and leave the library to these macros.

Rajan: No, we'd have to change the wording for STDC_VERSION to allow that since right now it covers compiler + library/headers.

Martin: We are adding a way to not conform. I think it is a problem, but we as a standard should not do this. Ex. __has_include. I support this in some non-normative way.

Lars: C++ added feature test macros for this already (library and language), and added a separate header for versions and predefined by the compiler.

Derek: Compilers have added flags to pass standards conformance test suites/validation.

Aaron: Would having this as an informative annex be sufficient?

Jens: Do you want this to be recommended practice?

Martin: What do implementations do now?

Jens: There are implementations of these TS functions already. Others will have to add in a number of things which is real work. This is a new thing.

Rajan: Wasn't Complex in C99 the same?

Jens: I want a straw poll for does the committee want feature test macros for each of the library clause sections?

Blaine: I want to see a sample of what this would look like.

Clive: Is it a test for each of the functions?

Jens: No, it is for the version of the header file. The changes are in the annex in the paper.

Philipp: Objections to having this normative. It is still useful for freestanding implementations.

Roberto: We are doing this due to the integration of TS18661. I know of compilers that define STDC_VERSION but never integrated Complex. How much time do we expect this to take for implementations to implement this?

Philipp: This is not necessary for freestanding implementations.

*Straw poll: Does the committee want feature test macros for each of the library clause sections as described in N2359?

18/1/5 (In favor, Opposed, Abstain)

Consensus for this.

If we have time later in the meeting, we will discuss the last two points.

6.6 Bhakta, The overloading of semi-colons in the C standard specification [N 2345]

OK with the change.

Leaving it to the editor to clearly change this in a way that make it clear.

6.7 Svoboda, Defining Undefined Behavior [N 2365]

Rajan: Perhaps do not do push/pop for this proposal?

Robert S: I don't think this is an accurate statement for –f{trap/wrap}v.

David S: Will look into it later.

Aaron: Any implementation experience with this? I think there will be implementation issues with this, especially any position ones (not once per translation unit at the start).

David S: I've listed switches on the command line but I'm sure there are pragmas. If a compiler doesn't support changing this in the middle of a translation unit, that is a programmer problem. Ideally this would be on a function level.

Derek: The cure is worse than the disease. People discover this when porting code, so they don't add them when writing the code. Pushing/popping will create a lot of problems.

David S.: It is a way for helping maintainers too.

Peter: I am in favor of the principle of the proposal, giving both the programmer and the compiler a way to say what they want or do. Specifically turning off strict aliasing doesn't turn off provenance based aliasing.

Lars: This seems to be introducing some kind of language dialects.

Peter: They exist now, but we don't have a way of talking about it now.

Lars: Now I can get it through third party code.

Niall: Most programmers write for their particular architecture and toolchain. They don't write to the standard. This is written the wrong way around. It should be a feature check. Ex. I write a static_assert for little endian, and so people trying in big endian would not have it compile.

Peter: Need something that asserts that it is guaranteed to trap.

David S: This proposal does make it more clear what the compiler will do for undefined behavior.

Freek: The Linux kernel is compiled with the optimizations off for problems like this. Would this proposal allow them to turn on optimization? I wonder about the usefulness of this feature.

Clive: This brings back C Safety and Security. Ex. No undefined behavior. Then someone objected saying "I know on my platform signed integer overflow will work". This will allow stating that explicitly so it is a good thing.

Derek: Platform profiles is the way to go. There is a history to do this.

Martin: The pragma mechanism doesn't really work. Cross function boundaries, link time optimization, etc.

Aaron: Need implementation experience. Ex. Here is a GCC fork with this implemented.

Peter: There is a list of flags for the Linux kernel that is used. Should we have these?

Martin: I would not be in favor of specifying dialects.

Peter: I want to nail it down.

Aaron: It is a lot of maintenance. Might be worth it, but it is a significant effort.

David S: Summarizing: Profiles per platform is not good since it doesn't work for undefined behavior, but does for implementation defined behavior. I have been trying compile the Linux kernel with Clang. I'm sure the kernel could benefit from this if they could move command line flags into a header file.

6.8 Seacord, Bounds-checking Interfaces: Field Experience and Future Directions [N 2336]

Rajan: 2.7: strerror does not take a programmer specified buffer. Function only takes one parameter.

Melanie: Does MISRA say anything?

Clive: It points to Martin's paper to say use Annex K with guidance.

Philipp: I like that the C standard is small and readable. This Annex K adds 10% to it.

Robert S: There is no evidence that these functions are bad. Regarding the size of the standard, I would actually get rid of the non _s functions.

Martin: It is valid I didn't have data. Cisco didn't go into the project with the expectation I would write a paper for it. The overall impression I got was that the experience was detrimental rather than positive. The constraint handlers code was all dead code since if it was wrong it would have been fixed (the original code). If the constraint handler was designed to abort, that would work.

Robert S: I would be OK with having it abort. Any runtime constraint violation is a programming error. Would that be enough?

Martin: There are important use cases for security functions. You can't have it abort in a transaction that could not be rolled back.

Robert S: An example where it makes sense is instead of testing inputs you could just use the constraint handling function. This paper tries to fix the problem to allow the constraint handler to be used in multiple threads.

Martin: This does not solve the general problem. It lends themselves to problems and incorrect use.

Aaron: Were you hearing about the problems only because you were the interface. No one talks about the benefits, just feed you the complaints.

Martin: I don't think so.

Robert S: The project you are talking about was where you had guidance to replace everything right?

Martin: No, there was no such guidance. There was no hard and fast rule, just guidance on development.

Robert S: Microsoft actually forces it. They have added it to government documents as well.

Martin: The other view is that no mainstream implementations have implemented this. Even Windows is not conforming showing that this is not something that is wanted or valid. Third party implementations don't use compiler intrinsics which is needed for performance.

Aaron: The library side: I know of at least two people who wanted to add this to glibc, but were told no.

Blaine: They may not know what they are asking for (the people that are asking implementations). I would like to see a paper for removing it.

Robert S: I think even Martin would want to see a solution before removing this.

Martin: No, I don't. There are better solutions out there already.

Philipp: Annex K has plenty of functions. It would be nice to know which ones are used.

Jens: If they were actually implemented.

Robert S: Some implementations have done the string functions and not the I/O functions.

Philipp: The wide character functions seem to only check that the pointer is not null which is not a lot of value add.

Robert S: One of the things I discussed with Richard Livingston is that they treat every error as a hard fail and log it. I hope to be able to study the information later.

Martin U: Is there any long term plans to do anything with strings like adding bounded strings?

Martin S: No plan, but anyone is free to write a paper for this.

Martin S: I would not want a straw poll for anything today.

Robert S: I would like to see if the needle is moving.

David S: I would like to see some changes to address Martin S's comments.

Robert S: There is some of that in this paper.

David S: N2225 is there to correct the threading issues with the set constraint handler.

Aaron: If the vote is no, is anyone willing to do the work?

Martin/Jens: Yes, we would.

*Straw poll: Does the committee want to remove or deprecate Annex K in C2X?

8/8/7 (In favor, Opposed, Abstain)

Blaine: Unless we see a concrete alternative, we'll stay this way. Martin, I take this as something to bring forward.

6.9 Douglas, Enhanced C/C++ memory and object model [N 2367]

Moved to Tuesday morning.

Niall: Updated and rewritten.

Jens: If you read some object the byte values will have the last stored values.

Derek: No, it is just the value that will be the same.

Robert S: For C, can't you access any object with unsigned char?

Jens: Yes, but the effective type does not change.

Robert S: Effective type is only for malloc right?

Jens: Yes.

Peter: All of the copying seems to be to work around reinterpret_cast. Shouldn't reinterpret_cast be fixed instead?

Niall: It wouldn't fly in C++.

Robert S: Does C++ have trap representations?

Niall: I'm not the right person to ask about this.

Niall: For C++, trivial copyable means the padding gets copied too.

Martin S: Why does this reattach function use the object directly but the detach uses a reference?

Niall: I'm not the best person to ask for this.

Niall: I want to get a sense from WG14 (even though this is not a problem for you due to type punning being allowed) whether or not this approach seems good for shared memory.

Jens: I think having undefined behavior is good since it gives latitude to do something like POSIX memmap to give defined behavior on top of the C standard.

Niall: This is more of a unidirectional cast. Input becomes invalid, output becomes valid. It can help give compilers and optimizers more tools for performance and also can be used by solvers to prove the program is correct.

Aaron: memmap is a solved problem. This is a way to reduce undefined behavior to reduce problems in user code.

Peter: If this was single program shared memory, all you'd need a release acquire fence, and no need for everything else.

Niall: Shared memory doesn't have cache coherency. Ex. Network memory. C++ gives magical powers to memmap.

Peter: The interesting case is pointer values read in. It doesn't have to happen during the memmap, it can happen on pointer read.

Robert S: What are you asking from us? Are you asking for us to change the language someway?

Niall: I'm getting the sense the room sees there is no problem for C. I was assuming a unidirectional cast was something that C would want, but I think I'm wrong.

Andrew: For my MISRA hat, I don't see what this is addressing or what problem it is solving within C.

Martin S: I would echo the last comment, but even for C++ I can't see what this would accomplish. Where is the instance of undefined behavior this solves that memory copy doesn't?

Niall: Non-trivial copy is what I expect C++ have changes to allow this.

Aaron: I like this in terms of helping with aliasing. Would it be better if WG14 said we don't have a problem with the code here?

Robert S: If it is a declared type, you can convert to unsigned char and change it. For allocated, it keeps the effective type.

Derek: For C, can't we have a pointer to a volatile of the appropriate type?

Jens: volatile doesn't help. The equivalent for C is to get rid of the effective type.

Martin S: The wording says character types are explicitly excluded from changing the effective type.

Jens: The first phrase says if it has a declared type it takes precedence. You can only change the effective type if it doesn't have a declared type.

Tuesday morning

6.10 C Memory Object Model Study Group discussions

Sewell, Exploring C Semantics and Pointer Provenance [N 2311]

Sewell, Moving to a provenance-aware memory model for C: proposal for C2x [N 2362]

Sewell, C provenance semantics: examples [N 2363]:

Peter: How does the committee want to work with this?

Derek: Where would you like to end up?

Peter: This paper doesn't talk about sub-objects, uninitialized variables, etc. but has something we want provisionally approve the text difference in N2362.

Derek: Do you plan on addressing the remaining bits?

Peter: Yes. You're going to get a better C standard.

Kayvan: We want to give a way to talk formally about the standard.

Derek: Some are interested in this, some are not. I expected something to help increase optimization.

Robert: I think there is something like an uprising in the marketplace by C programmers since what they see clear code giving different output from what they expect.

Derek: I am trying to balance the value of what is put into the standard with the benefit from those changes.

Jens: At some point it is a rewrite.

Aaron: Personally I see this is as a clarity aspect to the C language.

Derek: There is a higher chance of breaking things with more changes.

Clive: I want to reject this since I don't see a motivating case for this. The cases shown are really bad code.

Jens: There is a whole paper of examples.

Derek: But they are all edge cases.

Robert: Jens is taking an opportunity to improve the language of the standard. The result is more changes textually than intended maybe. Perhaps limit it to the minimum required for the new thing.

Derek: What is "improve?"

Martin U: There is no "new" thing.

Robert: If the same thing could have been done with smaller changes which would help.

Jens: There is not much that could have been taken out.

Philipp: There is small changes like the "storage instance" which could have been a way forward.

Peter: We had a clear vote for the way forward. For those who have issues, they can engage with the details and we are happy to do so between meetings. People need to pay attention in between the meetings because there is too much to do every 6 months.

Philipp: Having more advance notice before meetings would be useful for me to be able to attend. A week and a half is not enough.

Peter: A number of us are going to the C++ meeting in July. Being able to take info from here to there would be good for us.

Martin S: The benchmark for WG21 and WG14 has traditionally been implementation experience. In particular, GCC said this will invalidate existing experience and having to do this would be infeasible or very difficult in the current GCC framework.

Jens: I read that differently.

Paul: Was this discussion for new functionality or something different?

Peter: The idea was to have something consistent for everything but we can't since they are wildly different.

Martin S: I would be more comfortable with implementation experience.

Martin U: It can be changed based on the discussion here.

Peter: The fact that GCC has acknowledged bugs is not a reason to cast in it in stone.

Martin S: Joseph Myers has read it differently so not a bug.

Jens: For pointer equality the standard is clear.

Martin U: That's why I want more clarity. This way we can talk about it. We need to start with clarification then we can talk about changing it.

Philipp: But we are unclear on what the clarification will be different from what some think should be the way it is or what it currently says. It is implicitly making a change.

Derek: What is the interface between these models and the standard?

Peter: They would go in explicitly to the standard. The choices between the models is more how aggressive alias analysis can be.

David K: The basic idea is to track where the pointer was intended to point to. If you know that then you know when that pointer is out of bounds. That covers 80% of the problem. Then the corner cases come in. Ex. int -> pointer. The models handle that different ways.

Derek: There is a balancing act between the cost and benefit. It needs to be sold better.

Peter: We've already had a clear vote on the general direction. The study group will continue building on the models here, particularly the last one.

Robert: We took a vote on direction. Now we've come to the text, but reading the text I can't tell the difference between an object, storage location, and pointers. It seems some changes are done some places and not everywhere. I can't comprehend this as is.

Peter: The end notes have a lot of text describing it.

Martin S: I think WG14 expressed a preference for stricter rules. I would like to give direction to restrict code or implementations whenever there is a design choice.

Peter: We cannot have a design choice this way since you have to look for a wider thing.

Martin S: Constraining the optimizer has a cost.

Peter: Constraining code does too.

Robert: We have a value system that puts code first.

Martin S: Empowering optimizers also empowers analyzers and so it is better.

Roberto: You said Joseph Myers is against this.

Peter: No, he said this is good, but he has some interpretations of the standard that is strange.

Roberto: Can implementations be changed/extended to support these proposals?

Peter: I doubt they would do it speculatively.

Roberto: What are the GCC people implementing? What is Clang implementing?

Peter: They are implementing a confusing hybrid of different things.

Sewell, C provenance semantics: detailed semantics [N 2364]:

The goal is still to keep existing behavior and practice and make it valid. No new features or change to the language syntax.

Roberto: Pointer provenance is preserved even if it goes past the object.

Robert S: Does provenance affect memcmp?

Peter: No, it is in the abstract machine, not in the concrete representation.

Andrew B: The info in the abstract machine, having this information would be good for debugging in actual runtime too.

Robert S: You don't need to keep the size.

Peter: You need it to determine whether you have UB.

Martin U: You don't want arbitrary points to point to local objects.

Philipp: If the compiler can prove using the pointer directly again does not expose it would be useful.

Freek: I would like a straw poll where Derek said this is all nonsense and academic, to see if there is a problem here that the group sees.

Peter: I don't think we need a straw poll since we've had near full consensus in Brno.

Re pointer bit manipulations:

Derek: The standard currently doesn't allow the arithmetic operations on integer types and convert back to a pointer. It is UB.

Peter: Yes.

David K: You can't be sure this would work on all architectures.

Peter: Yes, this is just saying that if this worked on your architecture before, it will continue to work with this provenance model.

Martin S: My concern is the cost to implementations. They either give up or have to track all this.

Peter: Saying anything regarding provenance adds burden. The heaviness is up to debate. It is not clear this is a heavy burden.

Martin S: It is for implementations to keep its current degree of optimization and analysis or it has to give up.

Derek: What are the benefits?

Peter: Currently the problem is it is not clear whether compiler optimizers are valid. It makes what compilers are doing valid which was previous vague (not sure if it was allowed or not). It allows validation of compiler aliasing optimizations now since we have a model to test against.

Derek: Why not have a bunch of CR's for the problems?

Peter: This is an integrated solution. You can think of it as solving DR260.

Martin S: The proposal makes the inter-object integer arithmetic defined when GCC currently defines it doesn't. This will constrain implementations.

Aaron: The only thing you can do with ux is round trip. Any addition, even + 0 makes it now invalid and we're in UB land.

Peter: As far as we understand it, there is no proposal to make all compilers sound.

Jens: There is a big gain for application programmers to know what compilers will do.

Robert S: Reading a byte of a pointer doesn't make it exposed until you read all bytes.

Peter: No, we make any byte read expose the pointer.

Implications for optimization:

For p == q can hold in cases where p and q are not interchangeable.

Martin S: This is only possible with one past case. Making this undefined would make this better. C++ has specified this already without problems.

Jens: The standard already disallows the optimizations.

Martin S: But implementations do not follow the standard for this.

Freek: Why do you prefer UB over unspecified?

Martin S: Allows more optimizations.

Clive: This is like defining undefined behavior. If you could query your environment then you can say if X holds, then the rest of this follows.

Philipp: It would be confusing if one past pointers act different from regular pointers.

Peter: The explanation is that it is ambiguous.

Martin S: The examples are valuable but constructed in a way that is misleading. The data is not meaningful due to the compiler bugs they run into. We would need to generate more representative examples that don't suffer from the bugs.

Derek: How much code uses the features in the results table?

Peter: Making the compiler to be sound with this model would be a little work to make them more compliant with integer to pointer conversions.

N2362:

Freek: Does realloc create a new storage instance?

Jens: Yes. It was always the case in the standard even if the same address is returned.

Jens: Abstract address is the same as the uintptr_t if the implementation has it. It brings it from the conversion part of the standard to an earlier part. It helps with ordering and leaves room for segmented address spaces.

Philipp: Whether we need exposure depends on what model we use with provenance.

Jens: The proposal we have here is the most conservative we could do relating to the text of the current standard.

Do we want provenance, and if so, do we want exposure? Are we satisfied with the model? After that can discuss the one past pointer and round trip and other issues.

Roberto: We already have provenance; we are really asking if we want them in terms of this model.

Philipp: Storage locations seems useful independent of everything else. Perhaps just ask for that.

Martin S: I have parts I am strongly critical of, but overall I like the approach. I am not clear that we won't lose some normative text from these changes.

Jens: The group had a chance to look into that before this meeting.

Martin U: A way forward would be to split this up into things like storage instance.

Martin S: As an overall strategy, my biggest concern is rather than imposing stricter rules to constrain code out there which would cause problems for optimization and detecting bugs in programs. I would like to see the group more open to this type of feedback.

Jens: For me, I can't open up something I don't know. I want to find out what the state of the standard is. What you are saying is the next step. Once we know what is there we can talk about changing it.

Peter: Nothing we are talking about today is to make defined what is undefined.

Philipp: You are taking away some wiggle room.

Aaron: We are trying to fix a DR. I find implementation experience to be compelling, but since we don't all agree, the question about what we are affecting is not as interesting.

Derek: This seems to be a lot of wording to fix a single DR. It should be broken down into smaller chunks and list the problems as they are hit.

Martin S: We don't know what we are changing reflect what is there now or are becoming more strict.

Peter: DR260's resolution affects wide swaths of stuff.

Jens: One of the things where I broke it down is things like storage instance in the document.

*Straw poll: Is the committee OK with the general direction of the provenance proposal?

18/1/4 (In favor, Opposed, Abstain)

6.11 Gustedt, Introduce the term storage instance [N 2328]

Melanie: You said the object is the same as a sub-object but that is not true.

Jens: I want to have these changes in first to make it a two stage process, 1) get the space, 2) wrap the object into the space.

Philipp: The sub-object does not have a storage instance itself.

Fred: Did Larry object?

Jens: No.

Rajan: Did you talk to Larry?

Jens: No, but he sees all my proposed changes.

Martin S: I think it is a good change, but have nits on the wording.

Lars: I see sometimes replace space with storage instance, other places you replace it with storage. It needs to be looked at again.

Jens: I am happy to discuss nits.

*Straw poll: Is WG14 good with the general direction of N2328?

(15/0/0)

Clear consensus.

Jens: Bring issues forward to me.

Tuesday afternoon

6.12 MISRA Discussions

Banks & Ward, MISRA Liaison report and relationship with C Safety and Security Rules Study Group

Andrew: I am the MISRA C study group lead and official liaison to MISRA C. We quote the undefined and unspecified behavior annexes and have to pay a license fee to BSI. The MISRA 2012 first revision was published in February, and the next revision updated to C18/C2X will be sometime this year. We thank the C safety and security group for their very good review of MISRA.

Andrew (not speaking for MISRA): We may even end up publishing MISRA C as an ISO document even.

Clive: We've spent 2 years going over the MISRA document and need to see what we need to put into our C Safety and Security document. We don't want to hit a wall if we incorporate something from MISRA due to copyright.

Andrew: I'm not sure what the C Safety group is really doing other than giving a good ISO review of MISRA. If the scope was to unify multiple other documents, that would be fine.

David K: The scope is wider than that, it is to add in Safety and Security.

Andrew: That may have been the goal, but it is not what is happening. It is being critical of MISRA. The direction right now is wrong.

Robert S: The initial goal was to update the document to the later C standards.

Andrew: But how much of that has been done?

Robert S: None, we got bogged down. We ended up saying this is what we want to be the direction of MISRA. We didn't want to create a competing standard. We never got a settlement on how to work together. That was a drag on the study group. The two years have been truly study. In the early days of the group we had MISRA with the safety focus and the other people as the security focus and they were rarely aligned. Due to the conflicts there, the security guys dropped off, leaving it with MISRA.

Andrew: The resources MISRA put into this pulled away from MISRA C/C++ work. This made this a drain.

Robert S: My thought is this is now up to MISRA. My sense is the current effort should be dropped or go back to the security focus. If MISRA really wants to go all in and pool with ISO, then that would be the best thing.

Andrew: We categorically fully support the activity. But I question where it is going to end up. Unless we can refocus and define the objective beyond just taking MISRA's IP, then I agree we should close the study group. Note that MISRA is not a closed shop, so anyone can join. It really seems to be a duplication of our effort with

questionable benefit to the industry. Right now the study group is interfering with how MISRA is working.

Robert S: It really is your decision. If you want MISRA to maintain control of the standard, then you can say no, and the study group can refocus.

Andrew: I think the question is really whether WG14 wants the study group to continue.

Aaron: MISRA is for users, the study group is more for industry and analyzers.

Robert S: Does MISRA want to continue keep control through MIRA, or do they want to funnel through ISO?

David W: MIRA is just there since someone needs to own it. There are things for and against it. We have many stakeholders that we need to consult to see if this is something we want to do (go through ISO). Note that we are open to all for people who want to make significant technical contributions.

Derek: MISRA is a good document with a set of guidelines. Why does WG14 need to be involved with yet another document or guidelines?

Aaron: Because it targets static analysis vendors. MISRA is hard to work with. It is good for developers, but not for a tool.

Andrew: I am quite happy to discuss with you for areas to improve.

Robert S: I brought forward this update. This happened because I was asked by some auto manufacturers to merge the CERT guidelines and MISRA C guidelines. We agreed in principle to do it, but never got the details. I think we're at the stage we got to do something.

Charles: My take is neutral. I appreciate what MISRA has done in the past. My current industry is medical and we need both safety and security. I have seen unless it is in the standard itself, MISRA is seen as only automotive.

Andrew: That is not the case, not for a while.

Charles: It may not be the case, but it is the perception.

David W: The first implementation was in the medical space.

Charles: I agree, it fits there but it is not the perception. Having it in the C Standard is valuable where we can have people point to it giving it perceived legitimacy. Applying this to implementation is what really makes it work. Being able to put this into analysis tools is going to make this work.

Andrew: Redeclaring my interest working for LDRA. The ISO standard for medical references MISRA. The NASA guidelines are based on MISRA.

Philipp: Shouldn't it then be brought out of a private corporation and bring it to ISO?

Andrew: I don't care who or how it is done just that it is published. The original MISRA guidelines we published in 1994 and then republished as an ISO document that fed into becoming ISO 26262. Can it be fed into ISO?

David K: 2 mechanisms, fast track where it is owned by ISO or kept at MISRA and updates done by them.

Derek: ECMAScript did this outside the ISO route.

Clive: Whatever way to go forward, MISRA as ISO or the study group goes on. If the study group continues, perhaps one of the focus areas is where the conflicts are between safety and security.

Andrew: Addendum 2 and 3 are freely available on our website. There is a matrix showing MISRA C vs CERT. We found a case where MISRA conflicts with CERT (size of an array edge case). We have produced that delta. The claim that MISRA is just a safety standard is unfair.

Peter: What do you (Aaron) see the result of the document to be that would work for you?

Aaron: MISRA had a rule where you can't have two identifiers with names that are too similar to each other. This works for small projects but not for a large project with billions of lines of code.

Peter: For that sort of thing you'd want something that is decidable.

Aaron: Yes.

Clive: No, the MISRA rule was decidable. The MISRA rules are for developers starting with a blank sheet of paper.

Andrew: The rule is advisory and you can forget about it. A lot of us do.

Aaron: I think a different view from a tool vendor would be useful.

Andrew: I see your point but don't necessarily agree. We have rules that there because other standards require it like having a single return. Keep in mind more than half the group is tool vendors.

David K: Is MISRA going to make an IP claim against the study group?

Andrew: I think the question really is should the study group be disbanded.

David K: The study group can update to bring it in line with C17.

Andrew: We can update MISRA to C17 and forget about the study group document.

Robert S: We can continue dealing with safety and security but without MISRA, leaving it open to competing; Focus on security; Disband. It is more up to Charles.

David W: In terms of IP, we'd need a request and an agreement. We're more than happy to help. MISRA exists to help with that.

Robert S: My opinion is ISO couldn't care less whether we produce a standard. If they don't pay anything, would that end your deliberation?

David W: It is too binary; we are expected to pay ISO but I'm not saying either way. It needs to go through the appropriate channels.

David K: Since I don't hear a solution of the copyright issue, I suggest we keep the focus on updating to C17.

Aaron: Since most people are safety related now, do we have a study group?

David K: That is a question for the next study group meeting. i.e. Assuming no access to MISRA, what do we want to do?

Blaine: We have a couple of resolved DR's but it is not worth a new publication.

*AI: Charles to ask the C Safety and Security study group what they want to do under the assumption of not being able to use MISRA.

*AI: David K: Look into mechanisms for publishing the MISRA documents under ISO and discuss with Andrew Banks.

Banks, Follow-up on enumerating & cross referencing Annex J (formerly [N 2112])
Jens: Why don't you cite the place in the standard that has the behavior?
Andrew: Because it is not 1-1.
Robert S: We did create a numbered mapping in our TS.
Andrew: Previously commented: Having cross references helps having a sanity check on the counts (don't miss any). Another suggestion made up previously was having fixed numbers for the lifetime of an item, but that was not taken well.
Larry: The order is in the standard clause order. Echoing Jens. An informative annex is not a good thing to be referencing in another document.
Andrew: Even in MISRA we get asked where the numbers we added came from.
Larry: Adding things to the middle of the list cause issues.
Andrew: I am fine with whatever numbering scheme, you can change it, whatever. I am OK with changing numbers.
Blaine: The problem to me is that this is like a bibliography reference. Having a forward reference to the list would be useful.
Aaron: I know of at least 3 standards that tried to change this list to numbered lists. This to me says we need to number them.
Jens: I looked through it and the problem is artificial, and if there are places with multiple references, that's what it is.
Andrew: At least three groups have numbered it so they think it is useful.
Freek: You don't need to label with numbers it can be anything.
Andrew: Anything can be used. I am not tied to numbers.
Robert S: We can number them based on the section they appear. It can be appended by a letter if there are multiple reference in the same section.
Roberto: I presented a paper to remove one from the list. There are many that refer to 7.7.3.
Blaine: We can add in a special thing like notes that has a thing that labels undefined behavior with something like UB# which can then become the index of all undefined behavior.
Freek: I am good with doing the change.
Martin S: What is the purpose of the Annex? If we should be referring to the actual section, why have this?
Jens: For others.
Martin: Others have given feedback they want it numbered. What is the problem?
Freek: Is there any cross-referencing in the document already?
Jens: It is there already.
Aaron: WG21 has converted all bullets to numbered lists with a macro. Do we want to do the same?

David K: Do the editors have any significant opposition to this?
Jens: No.
Larry: No, I think it is misguided, but sure.
*Straw poll: Does the working group agree to uniquely label the bullet points in Annex J?
20/1/2 (In favor, Opposed, Abstain)

Banks, Follow-up on "defensive" attribute (formerly [N 2258])

N2258: Rajan: Minor typos (ex. Default in switch needs a ":").
Andrew: The real world has things like cosmic rays, memory corrupted by buggy code elsewhere, etc. This comes from a real-world situation.
Philipp: You can make your condition variable volatile and you have it.
Andrew: Doing that would break more optimization than this would.
Jens: The proposition doesn't give wording in the standard. How could I word it?
Aaron: N2365 talking about defining undefined behavior touches on similar issues that makes it hard to talk about it.
Andrew: I am only look into the principle of the thing, not specific wording yet.
Aaron: All attributes need to be able to ignore attributes. Since this cannot be ignorable, this should be a keyword or something else.
Blaine: I support using attributes here.
Martin S: What can an implementation infer about the two blocks of a boolean if statement? I don't see what the compiler can do in case that are impossible. The implementation needs to be able to reason about it. It cannot be that anything is possible.
Freek: You can have this done with a new volatile variable you can add just for this.
Blaine: The paper has this as a defensive mechanism for real world code.
Alex: Telling the compiler that code is reachable that is analyzed as being unreachable is nonsense. It does boil down to the equivalent of volatile where you can take the C++ volatile concept added in to do this.
Andrew: I put the attribute where it is to highlight where the problem is.
Niall: For a compiler, seeing any one of these would cause each branch point in the program to not be able to be folded.
Andrew: As soon as you take this away from the lowest level of optimization this becomes a problem in Clang and GCC from what we tried.
Blaine: I think it is doable from front end code.
Niall: Basically just turn off optimization to various degrees. That is what this is.
Peter: Optimization is not localized to a particular syntax location, so I don't know how to do this.
Andrew: I understand, but some level of optimization needs to be turned off.
Peter: But then you need to word this in standardese.
*Straw poll: Do we want to see something along the lines of N2258?

6/9/10 (In favor, Opposed, Abstain)
No consensus to move along these lines.

6.13 Stoughton, Proposal To Add Extended Month Name Formats to strftime() [N 2337]
Andrew: It is an interpretation request that has been widely reviewed and is going into the next POSIX revision.
Lars: Why was h was not used/added?
Philipp: Why is Ob/B switched from abbreviated and full name compared to regular b/B?
Lars: Changing O->E would break existing glibc implementations. Even though the man page says the opposite.
Larry: I don't see any problem with it being under O as well.
Nick: E is for a different calendar.
Geoff: We didn't add in h.
*Straw poll: Do we want to incorporate N2337 into C2X switching the cases for the b/B's?
22/0/4 (In favor, Opposed, Abstain)
Consensus. Goes into C2X.

6.14 Stoughton, Error Indicator For Encoding Errors In fgetwc [N 2338]
Aaron: Does this have the possibility of breaking code?
Rajan: This is like a DR. Can always break code, but this is obvious to me that this is a bug fix. Since this is an error case I expect if anyone does break it would be minor.
*Straw poll: Do we want to incorporate N2338 into C2X?
18/0/6 (In favor, Opposed, Abstain)
Consensus. Goes into C2X.

6.15 Stoughton, Change Request for fopen exclusive access [N 2357]
Rajan: Second change is better.
Jens: I have problems with the wording of file permissions showing up in the first change. We don't mention that in the C standard.
Geoff: That wording is already used in Annex K.
Niall: I am in the C++ low level file I/O subgroup so I can talk about this with authority. I would not have it say what it should not be and instead talk about what it should be. You should be able to atomically create a new file and the standard should be changed to say this.
Rajan: Any objections to the second change? There are other file systems beyond what was said before for other systems.
Blaine: Love the second change, hate the first one.

Robert: This wording appears thought annex K so we should change it everywhere and not in isolation. I would like to see Niall's C++ proposal here.

Niall: I expect it is not very relevant here. I expect to bring it to C in about 5 years' time.

Niall: The second form allows doing nothing. What is the chance of existing code breaking since you don't have to have a lock file where now you can get corrupted data?

Geoff: The exclusive use is separate from the creating a file and was always like that.

Niall: I was talking about a lock file not locking the file on creation.

Nick: Implementation defined was deliberate to allow specifying what POSIX does.

*Straw poll: Do we want to incorporate N2357's second proposed change into C2X?

11/1/11 (In favor, Opposed, Abstain)

No consensus.

Blaine: I object to that; abstentions should not be counted.

Aaron: I thought consensus was up to the convener but ISO had guidelines.

*Straw poll: Do we want any changes to 7.21.5.3 p5 regarding the "exclusive" in exclusive file access?

11/1/12 (In favor, Opposed, Abstain)

David K: Does anyone object if there is some kind of change from the people who abstained?

- No response

Niall: I can propose alternative wording. I intend to write exactly what POSIX does.

Geoff: What you are saying is striking the second sentence in paragraph 5 is enough.

Rajan: Removing that is not good since it makes existing implementations not conformant if they don't support exclusive mode.

Wednesday morning

6.16 Integrating floating-point TS updates into C2x

Thomas, TS 18661-1 plus CR/DRs for C2X [N 2314]

Thomas, TS 18661-1 plus CR/DRs for C2X with change marks [N 2315]

Thomas, TS 18661-2 plus CR/DRs for C2X [N 2341]

Thomas, TS 18661-3 as annex [N 2342]

N2374 (slide deck)

Martin: Annex vs in body of std.

Jens: Likes as Annex.

Dislikes N in types.

Rajan: Similar to existing PRTN

Jens: Too many functions; might clash with existing code; wants renames with reserved prefixes.

Rajan: Will consider renames for parts 1, 2, and 3 as WG14 directs.
Aaron: TS experience not be that important when we roll into std.
Martin: Link errors with name clash?
Rajan: If user has own function with same name as std name, there may be name clash at link time.
Martin: Need a general solution for so many new names from all parts.
Jens: Not just a math issue.
Rajan: Already voted in; this is just a heads up.

Thomas, C2X proposal - TS 18661-4a [N 2355]
N2373 (slide deck)
Philipp: Why need these?
Rajan: Performance and accuracy.
Alex: Functions express intent of user (compilers may not do transformations, e.g., pow() to pown()).
Aaron: Did study group look at name clash with user code?
Rajan: No.
Keaton: If we subset, users will ask why we did not do all of IEEE-754 functions.
Jens: Need to solve name space; use prefixes.
Philipp: Use a new header to new functions.
Martin: Too many names.
Rajan: IEEE did not add in very specialized functions; just added general functions.
Keaton: Name space issue is more than just math.
*Straw poll: Should we adopt 18661 part 4a (not reduction functions) into C2X (not consider naming issue)?
(10/1/7)
Result: Goes in.
Jens: I want to resolve the naming issues before voting this into C2X.
Philipp: There are a lot of floating point programs that are OK with the set they are using but could have problems if they get new ones. It may make sense to have a separate header.

Thomas, update for C2X payload functions [N 2356]
Fixes issue of 'unsigned' raised by WG14 at previous meeting. Changes return value for errors.
*Straw poll: Should we adopt N2356 into C2X?
(13/0/5)
Result: This goes in.

6.17 Tydeman, FE_TONEARESTFROMZERO w.r.t. FLT_ROUNDS [N 2319]
Derek: You've changed 1 in FLT_ROUNDS to be ties to even. Is this a problem?

Fred: I don't know anyone who does otherwise.
Fred: You need this new rounding mode for nextafter for double double.
*Straw poll: Do we want N2319 to be put into C2X?
(12/0/6)
N2319 goes into C2X.

6.18 Tydeman, Precision and NAN/NAN(...) and INF/INFINITY [N 2320]

Fred: This will break every implementation. I am not proposing this anymore.
Rajan: Like the alternative.
Martin S: I don't see how this fixes anything. Existing programs will not be safe with this change.
Roberto: This allows a way for a program to set a limit for new programs.
Jens: I would object for the naming choice.
Fred: Not tied to the name.
Martin S: New code can use snprintf.
Rajan: Having a limit does solve the issue from a programmer perspective and analyzer even for existing code.
Philipp: I prefer making an explicit macro, not just say we have a limit.
Martin S: The precision idea was not viable. I think the problem is still there and a solution needs to be found like what was there in the original proposal, even though the committee decided to do precision.
Fred: Has anyone asked for a control on the output.
Rajan: No. No requests.
Philipp: If we do provide the maximum macro, we should have a sentence that the value could be zero in case the implementation does not support it.
*Straw poll: Do we want something along the lines of the macro addition part of N2320 to be put into C2X?
(9/2/7)
Consensus. Result in asking for a new paper for better wording for this.

Wednesday afternoon

6.19 Tydeman, Nextafter/nexttoward/nextup/nextdown [N 2321]

Fred: The chairman for IEEE has a use case for the existing words for this so I am withdrawing this change.

6.20 Tydeman, SD3 9: PreProcessor unspecified line numbers [N 2322]

Derek: There is only explicit unspecified behavior, no implicit.
Fred: I do not know.
Fred: I am explicitly making it unspecified.
Derek: It is probably in the original c89 mailing list somewhere for the original DR.
Fred: No implementation does it all. They do parts.
Derek: You need to enumerate the choices.

David K: It is there. It says “which of those lines”

*Straw poll: Do we want N2322 to be put into C2X?

(13/1/3)

Consensus. N2322 goes into C2X.

6.21 Tydeman, SD3 1: DR 440: Test macros for FP being 754 types [N 2323]

Fred: Jim said the extended types don’t have limits.

Philipp: Why is there no LDBL versions?

Fred: The original text only has macros for FLT and DBL, not LDBL.

Roberto: Are there implementations that could define this as 1?

Fred: Jim also said we should not have operations and values.

Roberto: Flushing to zero is programmatically set for sub-normals.

Fred: No, not always. Some hardware has switches others do not and always flush.

Fred: I can write a new paper removing the extra verbiage about operations.

*Straw poll: Do we want something like N2323 to be put into C2X?

(8/1/8)

Consensus. We want something like this in C2X.

*Straw poll: Do we want something like to option 1 or option 2 in N2323 to be put into C2X?

(9/0/9)

Consensus. We want something like option 1.

Fred sent an updated paper with the operations part separated out as per Philipp’s request (resolution for DR440).

Martin S: Process issue about forgetting or leaving something for defects in the existing standard.

David K: We’ve always had the requirement that someone champion the paper all the way through.

Martin S: If there is a serious issue and then someone who worked on it drops out, what happens?

David K: Then it must not have been serious enough if no one wants to pick it up.

Martin S: The process is not good enough. We should have a list of these things.

David K: You (Blaine) could add this to the end of DR440.

Blaine: Perhaps we need something with DR status’s.

Robert S: We’re trying to solve a problem that’s been solved for 30 years: Defect tracking systems.

Philipp: Don’t we have an issue tracker in our git for the standard?

Jens: No, we don’t have one. Can we do what the Austin group uses?

Martin S: I volunteer to maintain that list that lists documents and what we did with it.

Leave this update until we get an N# or tomorrow.

6.22 Tydeman, SD3 13: DR 482: Macro span files: undefined [N 2324]

Derek: Why not make it implementation defined instead of undefined?
Aaron: What does it mean to have undefined behavior in the preprocessor?
Jens: Some have undefined, some don't, some have 'shall' in the paper.
Fred: Yes that was intentional.
Aaron: If we change this I would prefer making it not allowed rather than undefined.
Aaron: What does C++ do?
Fred: I don't know.
Roberto: This is similar to the final line in the file not needing to be ending in a newline.
Aaron: C++ makes this an error. Not allowed to have a non-terminated comment, …
Fred: This is neither. This is a macro invocation.
Niall: There are 2 Boost libraries that use meta-programming with preprocessors. I would suggest we ask him what we should do and do what he says for C and C++.
Philipp: If undefined behavior is not liked, we could make it implementation defined.
Derek: So we're going to break conforming code to make non-conforming compilers conforming.
Aaron: C++ still does disallow this.
Fred: No, it does since that disallowable is in a preprocessor directive.
Fred to follow up on this. New document N2379.

6.23 Tydeman, SD3 11: Maximum normalized FP number [N 2325]
*Straw poll: Should N2325 be put into C2X?
(13/1/7)
Consensus to put N2325 into C2X.

6.24 Tydeman, Merge DR 432+467 [N 2326]
Derek: Is -0 really another kind of floating point number?
Fred: Yes, this is not required in the model.
Philipp: I am not comfortable with adding 0. Are there implementations that do not have 0? What is the cost for that?
Rajan: The paper was in the mailing, implementations could have spoken up if we have broken them, or talked about that as well. In this case we do have it in the standard that zero is talked about in other places so if they didn't have 0 they would have been non-conformant anyways.
Fred: This has been available since 2013 (the original paper).
Derek: What is the definition of 0.0 in bits?
Jens: 'if' statements select on zero so they would have to have a way of distinguishing between zero and near zero.
Larry: That is not true that the model does not have zero, if the fraction bits are all zero then we have a non-normalized zero.
Derek: What would happen if we didn't do this?

Aaron: The two DR's would not be resolved.
Philipp: Is that zero a mathematical zero or not?
David K: If it was a value, it would have been in code font so it is the mathematical zero.
*Straw poll: Should N2326 be put into C2X with editorial changes on the footnote expected?
(14/1/6)
Consensus to put N2326 into C2X.

Thursday morning

6.25 Gustedt, intmax_t, a way out [N 2303]
Derek: If I am a compiler vendor and you are a system vendor, and you decide you have a 128-bit type I don't need to support it.
Jens: GCC does this, they have a type that is __int128 for example, but it is not portable beyond that.
Derek: But this is a case of who controls intmax_t.
Jens: No this is about people fighting about ABI's.
Derek: But as time goes on, larger and larger new integers can make intmax_t the smallest type.
Fred: Is long double _Complex a semantic type?
Jens: This is about integer types.
Niall: When we talked about this before, we wanted to keep intmax_t the same, but decided on passing the extended integers as a pointer so not by value.
Blaine: Why not fundamental types?
Jens: My proposal is beyond that. Not just fundamental types. If you make it as that, it loses its usefulness.
Derek: Your change make it lose all meaning too.
Blaine: A RISC architecture has 128-bit integers as a fundamental type.
Niall: What about 8000+ bit types? No one will pass these by value, they will pass it by pointer. The committee should be aware of that and make a solution that handles larger types that should not be passed by value.
Philipp: There are implementations that have a 24-bit integer type. What if 128-bit proves to be so useful that we want to make it a basic type? That would break the ABI still.
Jens: If we change that we have many other problems since a new C standard could not claim ABI conformance.
Martin U: There is no fundamental type in C.
Martin S: Yes, that is a C++ term. For C it is basic type.
Derek: Why not have version numbers of intmax_t like intmax_t2 to keep the ABI the same.

Martin S: This standardizes existing practice, right?
Jens: No, this go beyond that.
Martin S: This doesn't break existing implementations though, right?
Jens: Yes. Correct.
Philipp: Do we know if C standard library functions are used with intmax or not?
Jens: printf is the one I know is used.
Alex: I think it is overly proscriptive to force by reference passing of larger integer types as Niall suggested. Perhaps not guarantee single copy semantics.
Robert: This is a problem. Could we eliminate the type? How painful would it be?
Jens: If you are good with that, just 'long long' for any new architecture. Then the preprocessor description text would have to change.
David K: There is a historical reason for intmax_t. You could always cast to long then 'long long' but we knew that may not be big enough so we added intmax_t. Why does GCC need this?
Robert: The problem with intmax_t is that it is the largest of the smaller integers.
Jens: Having this would allow GCC to use these with PRI* macros, etc.
Derek: Why not create a new type like intmax_compiler_t? Aren't you solving the wrong problem?
Jens: If you want to write a proposal for this, you can.
Blaine: intmax_t represents the largest integer fundamental type that is the fastest for the architecture.
Niall: va_max_t would be for the max to pass through vargs, and then eintmax_t for the largest extended integer types.
*Straw poll: Does WG14 agree to have extended integer types that are wider than [u]intmax_t?
(16/1/4)
Clear direction.
Martin S: I find the proposal difficult to read. There are typo's in the proposed changes. I see this as a first step, but need it to handle the problems like what David K brought up like how to print int128, int256 or whatever is the largest one?
Martin U: I see two problems: How can you find the largest extended type? For printf, need to be able to pass in a reference to larger integers.
Aaron: Do a compare and contrast of these solutions would be useful. Why you would choose one over another is useful to me.
Blaine: Bignums have been around forever so having the pointer to it as a general solution would be a general thing.

6.26 Gustedt, Clean up atomics [N 2329]
Blaine: The _Atomic had problems and when we introduced it and we knew it so this is a good thing.
Blaine: The real problem for this is extending this to floating point.

Jens: Yes, there are examples for this in the standard already and this is designed to work for integers, floating point types and pointers.

Blaine: I would prefer this atomics clean up first to handle the thread clean up later.

Blaine: Is there a binary compat issue with the selection rules for generic functions?

Jens: No, I don't think so.

Philipp: Compound operators based on the atomic fetch op's, why do we have the ones without the explicit?

Jens: Shorthand and since they are different (return the value first). I think with these we can be more particular for floating point things without saying generally "nothing bad happens".

Martin S: I have the same observation as the previous paper. It is better if it was broken down.

Jens: The alternative would have been to have 20 papers.

Blaine: The committee said they wanted it as a single omnibus solution for this. It is needed since the issues are interleaved.

Martin S: I am not opposed to making changes along these lines but I prefer separate papers.

Robert: We had a conversation before that the English text does not match mathematical model. I'm not confident the text we're modifying should be the starting point.

Peter: I don't think Jens changes here deal with the core concurrency model.

*Straw poll: Is WG14 happy with the proposed non-normative changes (section 2.1) in N2329?

(14/2/5)

Result: Good with the changes.

Martin S: We never lump in a number of separate changes lumped together for a software project and yet we are taking that approach here. I am not saying these are bad changes but I don't understand what we are actually voting on.

Blaine: You had the paper before with everything there.

Robert: But we didn't have the polls in sections.

Blaine: This is what you asked for.

*Straw poll: Is WG14 happy adding into C2X the proposed additions for the synchronization of the library functions proposed in section 2.2.1 of N2329?

(10/5/5)

Not clear direction.

Jens: I will make a separate paper.

*Straw poll: Shall we use the atomic_fetch_OP_explicit functions as a model definition for all the other compound assignment operations as proposed in N2329?

(10/1/9)

Jens: More clear, but some more convincing to do.

*Straw poll: Should we introduce the new atomic_OP_fetch[_explicit] operations as complement for all OP= operators and adding ordering as specified in section 2.2.3 of N2329 into C2X? (This would give the value after the operation)

(7/8/6)

Result: No consensus.

Martin S: Does this fix anything or just for a complete set?

Jens: For a complete set.

Niall: All code I read does the operator then fix up afterwards. You can already write operator += right now, so why do we need a new function to get the same thing.

Jens: This is to complete the model in both directions.

Niall: I get that but I don't know why atomics need to be the same as arithmetic. They can special.

Jens: This is different between C and C++.

Niall: C++ always had OP=, so that is not the case. Since you could already do it I don't see adding this a being valuable.

Philipp: This is adding more to the standard to read for relatively little gain.

Jens: We do this in other things in the standard already.

Philipp: Some of these functions are exactly the same as existing functionality so there is no need for them.

Jens: For pointer types and consistency.

Martin S: We should not take this a definitive no.

Jens: Yes, I take it as I need to write another paper.

*Straw poll: Do we want to add in the selection rules for atomic generic functions as given in section 2.2.5 of N2329 into C2X?

(9/4/8)

Result: Not clear.

Robert: Note that 2.2.5 lists the general idea, but the changes are in other places throughout the document.

Jens: I will write another paper.

Niall: For out of band error bit since it breaks ABI, can I get a yea-nay since that would change what I need to write.

Jens: It does not break ABI, so we need to talk.

6.27 Gustedt, Moving to two's complement sign representation [N 2330]

Robert: Width macros are brought in by the TS. Does this mean I need to add the floating-point header?

Jens: No, it is integrated in the right place (Fred: limits.h).

Rajan: Making extra bits into padding bits can break existing implementations (ex. If they truncate or anything else other than pad). It can change structure layouts thereby.

Derek: The other operation common in graphics is saturation. It can be done here too. We don't want to be too slavish following C++.

Jens: We will likely need to change the shift behavior in the proposal (for right at least) due to comments from embedded (Philipp and Derek).

Robert: For left shift we already voted to not make left shift undefined so it is OK to be ahead of C++.

Blaine: I am uneasy with the adding padding bits part.

Jens: I am not happy with that either but I want to keep header compatibility.

Philipp: Some places for right shift take 7x as long as logical so it is not good. Is the possibility of having padding bits used? If so it may be OK to be different from C++.

Derek: Apart for wrap, saturate and signal, what else can chips do for arithmetic overflow?

Jens: I hear that we are uncomfortable with these changes so I see that.

David K: -INT_MAX -1 may be a trap representation in certain implementations.

Fred: I know of one old complier that does it even though the hardware supports it.

*Straw poll: Does WG14 agree to fix the minimum value of signed integer types to the negated power of 2 of the width - 1?

(16/0/4)

Clear direction to fix the minimum value.

Martin S: In general I prefer smaller proposals rather than big omnibus ones.

Jens: If I did that, it would take more time and effort and they build on each other.

Aaron: In WG21 we had the same concern but without a holistic view we hit corner cases.

David K: What Martin is saying is a really important principle to break proposals down into separable concepts. It works a lot better if it is broken down and in the long run it gets it done faster.

Jens: That is no problem at all. I can throw 20-30 papers at you. I can automatically generate papers for each patch even but we'd spend all week on the papers.

Roberto: You insist that the bit-field changes is a single package with the padding bits?

Jens: Yes.

*Straw poll: Does WG14 agree to proposed changes for bit-fields as specified in N2330?

(7/7/6)

No consensus.

Derek: I think the conversion changes are close.

Jens: I will take enumerations as a separate paper.

Philipp: BOOL widths seems weird.

Jens: For consistency and for token pasting reasons.

Philipp: I want either bool width out or bool max in.

Martin S: There seem to be mistakes in the diffs here too. In one case you have value of a macro being 0 instead of the expansion of a macro being 0. This is a normative change.

*Straw poll: Does WG14 agree to the general direction proposed for specifying the bounds of integer types based on widths as specified in N2330?

(20/0/1)

Clear direction.

6.28 Gustedt, Unify string representation functions [N 2360]

Philipp: Why do we need these functions at all? We already have printf.

Jens: I agree, these came in from the FP TS's. I also think we need to give constraint violations for things that are bad like the format string in my proposal if it is not a string literal. GCC already warns on this.

Aaron: I like this. Type generic is consistent. The string literal format string is good in that it removes one security issue, but it doesn't work with internationalization.

Jens: This does not cut off locale. It is only an effect for the decimal point. My idea is they are the C locale.

Clive: I generate format strings on the fly. It is also a lot more names.

Jens: We can throw out the functions entirely and only have the type generic interface (so implementations can call the functions internally if they want). For your use case, is it important to be type generic?

Clive: I wouldn't be able to use these functions. I'd have to fall back to printf.

Lars: I prefer to not have the string literal to be there. C++ added functions like this called tochars, intended for speed and low level.

Jens: If you provide the format in the string literal, you get that, otherwise you get a default. It is not intended to be slower since it can be done at compile time.

Lars: I want a harder look at this to be closer to C++ tochars.

Jens: The advantage here is for C library implementers this is easy.

Lars: I am afraid of that since it is slow.

Philipp: You can optimize this.

Martin S: You can't due to locales for floating point.

Jens: Beside that issue everything can be done at compile time.

Martin S: GCC has the optimization and it does not have to be a string literal.

Jens: I don't know how to formulate or implement it easily.

Aaron: There is a security benefit for tainted input.

Martin U: That is more of a user problem.

*Straw poll: Does WG14 like a type generic interface to print all basic types into a character buffer as specified in N2360?

(12/1/2)

Clear direction to do it.

Martin S: You can just pass integers for width, precision, etc. so don't need string literals.

Jens: My personal preference is to allow the user to select between %A and %G.

Aaron: C++ uses an enum to select.

*Straw poll: Does WG14 want to restrict the format of the tostr functions to string literals?

(3/5/6)

No consensus.

Rajan: This is something new in the C Standard with overloading with a different number of parameters.

Martin U: Do we want macros that expand into something to be discussed first?

*Straw poll: Does WG14 like the default argument format method of overloading for the tostr functions?

(9/2/4)

Consensus to allow default arguments.

Jens: I want to know about getting rid of the FP functions.

Martin S: But you couldn't now take the address of the functions.

Martin U: This changes how people think about C. Perhaps have these stand out visually or something.

6.29 Gustedt, Out-of-band bit for exceptional return and errno replacement [N 2361]
6.30 Gustedt, Align spelling of keywords with C++ and make them feature tests [N 2368]

Rajan: It does affect user code.

Philipp: I am concerned with nullptr in particular.

Aaron: I don't like having nullptr being a macro. I want it to be a full class object. I don't want it to be always a void pointer.

Jens: To be clear, I want it as a keyword, but also a macro.

Aaron: For noreturn, C++ defines it as an attribute so we can consider that which may interact with this paper.

Jens: Doing macro expansion on all of this is not fixed.

Martin S: Aestically I like it, but I am concerned about breakage. Note that _Atomic is still there so we don't really get rid of the "ugliness". What do you see going forward for new keywords?

Jens: I see this for the _DecimalN types being added into C2X. My idea would be to do something the same way for that.

Blaine: You can talk about different approaches for implementers in a new paper explicitly.

Martin S: I see nullptr as a separate thing and should be proposed separately.

Alex: I don't like how C added _Noreturn, and the addition of _Complex. I like making this more C++ like.

Jens: Hans B. has been talking in the reflector for _Atomic. For _Complex I would like a sharable mechanism between C and C++.

Niall: I think for static_assert we should have a 2 arg form.

Jens: We already have that.

Roberto: There is a bug in nullptr since it removes the possibility of having 0 casted to a null pointer. We can talk about this later. Also in the changes list on page 7 you see static and static that are the same.

Alex: You left _Generic off the list. Was there an intent to do this? Not doing it all it may imply something to the user on how they are used.

Jens: There was never a macro to replace it. If you think it is a good idea, we can get a new paper for it.

*Straw poll: Does the committee want to go in the general direction proposed by N2368?

(17/2/2)

Clear direction to move forward with this.

Philipp: We can have a first proposal for the non-controversial items.

*Straw poll: Does WG14 agree to the following list of keyword and macros as being subject to N2368:

alignas, alignof, bool, static_assert, thread_local, false, true

(19/0/1)

Result: Clear direction for these items.

Thursday afternoon

6.31 Sebor, Toward more efficient string copying and concatenation [N 2349]

Philipp: The last character may not be a character or have a one past the end pointer.

Martin S: The wording comes directly from POSIX. I wouldn't want to change it but we can tweak it after if we agree with the idea.

Fred: Is there a reason there is no wide character version?

Martin S: I am focusing on the narrow functions now and testing the waters now with this for incorporating POSIX functions.

Jens: There are a lot of things in POSIX that are in use and are efficient so we should look into it.

Derek: Are we going to require hosted and non-hosted systems to support it.

Martin S: No, right now just for hosted.

Philipp: Jens asked for the size argument first.

Jens: If it was a new interface then it should be, but since this is existing in implementations, I am fine with it.

Aaron: Do we know how many CV's are out against this function?

Martin S: It is a pretty obscure function, but I haven't seen any. It is still useful.

Aaron: If there were security issues, I would like to deviate, but if not, it's fine.

Fred: Do we want to put in Recommended Practice to have people use this over memcpy/strcpy.

David K: It is better than strcpy since it is more secure with the addition of the length.

*Straw poll: Do we want to put N2349 into C2X?
lots/0/1 (In favor, Opposed, Abstain)
Result: Goes into C2X.

6.32 Sebor, Defining new types in offsetof [N 2350]

Martin S: I would have preferred an optional diagnostic but C has no way of doing that right now, hence this proposal. The precedent would be C++.

Aaron: Conditionally supported with implementation defined semantics is how C++ supports it.

Martin S: I saw this as ill-formed but no diagnostic required.

Martin U: It would be better if we could solve these problems in the preprocessor or something else.

Martin S: I think that is fine, but no proposal for that.

Jens: I am in favor of doing something here. This is implicit undefined to explicit undefined is good. Another way would be to have offsetof being a real construct other than a macro,.

Alex: If the struct came after the member name would have eliminated this problem.

*Straw poll: Should N2350 be put into C2X?
(14/0/1)
Consensus to put into C2X.

6.33 Sebor, Add strnlen to C2X [N 2351]

Robert: One possibility is (since we have too many string functions according to some) to promote some functions from Annex K into the non-optional parts of the standard?

Martin S: I don't have a problem with promoting some functions in some form. Overall as an idea, reviewing a subset of Annex K and bring it in would be doable.

Jens: FYI, the _s function doesn't use a constraint handler so that is good.

Robert: Perhaps redefine the original function instead to return 0?

Aaron: Are you expecting to later add the wide char version since I don't want to forget and have an inconsistent state.

Martin S: As before I specifically chose to focus on the narrow character set. It is worth considering. I want to know if we want more functions along these lines like the wide character ones.

Aaron: I would vote for it if we had both wide and narrow. Otherwise no.

Philipp: There are some operating systems (Windows) that have the wide character definition to something useless. I wouldn't want to just throw in the wide char functions in by default.

Aaron: The wide char functions are used widely on Windows since the narrow functions have different code pages.

Clive: The description of the return doesn't make it clear the function will return maxlen if s is NULL.

Martin S: The wording is from POSIX and doesn't explicitly say anything about NULL.

Philipp: This is the way the whole section is.

Fred: Is the word bytes supposed to be changed to characters?

Martin S: Yes.

Clive: Each function in Annex K starts with the statement that the parameter is not NULL.

Martin S: I would strongly not want to have one string function with something about NULL as a parameter but not others.

*Straw poll: Should N2351 be put into C2X?

(11/6/6)

Not clear consensus.

Martin S: What else would people want to see be introduced into C2X in the future from POSIX?

Robert: Things from TR24731-2.

Aaron: You probably have access to a large body of code so why not look to see which ones are used often?

6.34 Sebor, Add stpcpy, and stpncpy to C2X [N 2352]

Philipp: I don't know if the zero filling is that useful. Also the memccpy is more better/general. Also these functions are not in the reserved namespace.

Martin S: There is little risk regarding the name. I don't want to break the expected symmetry.

David K: I like the idea but they are almost but quite a good fit since stpcpy doesn't allow a length and stpncpy negates the length benefit by the performance hit of zeroing.

Martin S: I agree, I want to add these to aid in porting code to non-POSIX systems. GCC looks at usage patterns and can transform code to more efficient stpcpy calls from user code using strcpy.

Philipp: Wouldn't memccpy be sufficient?

Martin S: It could happen, but it is not as nice to use.

Robert: Perhaps come back with this again addressing David K's comments and the namespace issue.

Martin S: I am not in favor of that. I don't want to invent.

David K: For compiler transformations, the functions don't need to be in the user namespace.

Martin S: We can have a compiler runtime library, we don't yet.

Clive: I take it removing the undefined behavior if you pass NULL argument still applies?

Martin S: Yes.
Aaron: Name reservation is not compelling to me since users don't know that.
Robert: Do compilers check this?
Aaron: Not for these names but do for _X to avoid getting a ton of bug reports.
Larry: strn* functions are useless in general use. I don't want those. I would like the wide character versions though.
Jens: I would like stpcpy but not the n version.
Martin S: I agree with Larry, strn* causes no end of problems for misuse. But I would not be in favor of splitting these up.
*Straw poll: Should N2352 be put into C2X?
(2/9/11)
No consensus.

6.35 Sebor, Add strdup and strndup to C2X [N 2353]
Philipp: Could we take out the sentence that sneaks in strnlen?
Martin S: Typo.
Aaron: Is it possible to get a non-null string from this function?
Martin S: It says it.
David K: I will note that strdup is the most useful POSIX function I have used.
Martin U: The issue with the size of the buffer in strndup was removed?
Martin S: No, it was elsewhere. We can add it in as a footnote if we wanted.
*Straw poll: Should N2353 be put into C2X?
(15/1/5)
Consensus to add it into C2X.
*Straw poll: Does the committee want a proposal for the wide character versions of any POSIX functions voted in this meeting?
(12/2/7)
Consensus.
Fred: The POSIX standard says it copies in n + 1 characters.
Clive: We do say we add in a null.

6.36 Sebor, Constraints on parameters to main [N 2354]
Derek: There is nothing preventing compilers from emitting messages for anything.
Martin S: Is it valid to call main with argc being -1. The standard is not clear. When are these intended to apply?
Larry: I think since they are in the program startup section it answers the question.
Philipp: I agree the current wording is not clear, but the constraints should be only on the program startup.
Robert: If we don't make the decision that here we can restrict the diagnostics that can be emitted.

Martin S: Yes, it is certainly possible a program can handle this or have an error. It would be helpful to diagnose this.

Aaron: Do we have a constraint that said argv can't alias anything else?

Martin S: No.

Peter: The argument that diagnostics are useful is true if and only if the code that calls main itself does it in a way that violates these constraints.

Jens: How much code uses main this way?

Martin S: A good question, I am not sure. An optimization was made for this.

Peter: Among the large number of things that are not clear in the standard, this is minor and should not be high on our agenda.

Martin S: You are questioning my priority?

Niall: We all have had the thought as a younger programmers to have main fix the arguments and then call main again with the right arguments. Older programmers realize they can call another function so we should just ban this.

Robert: That could break more code.

Martin S: That would be a loud breakage instead of silent breakage.

Melanie: The defect report said we don't want to see anything more on this so why are we?

Martin S: This is different, it's about the constraints.

*Straw poll: Do we want to put N2354 into C2X?

(5/11/6)

No consensus.

*Straw poll: Does the committee want to ban calling main recursively for hosted implementations?

(12/8/1)

No consensus.

*Straw poll: Is it intended that the constraints in 5.1.2.1.1#2 apply only on the initial call to main?

(14/2/6)

Consensus that the constrains only apply the first time.

Friday morning

6.37 McKenney, Pointer lifetime-end zap [N 2369]

Derek: On storage limited systems you may want to reuse the storage used by the pointer.

Melanie: On slide 5 you said load, did you mean the pointed to object?

Peter: No, loading the pointer.

Martin S: It is not just this, could be diagnosing things like members in compound literals that are used when it's lifetime ends. These opportunities are useful and would not work if the pointer could keep its value.

Martin S: Why is option 4 impractical to find all the end of lifetime code? It could be done with better static analysis or compilers.

Paul: Progress in this area is slow. It is a hard problem.

Derek: The issue is what to do after you find them.

Derek: Currently for the example in Paul's slide just works?

Paul: Yes, right now it just works since compilers don't stop it from happening.

Martin S: What would be the code change needed to make this code not rely on undefined behavior?

Paul: You could have locks.

Martin S: If you operated on types other than pointers then it would be performant.

Paul: Yes, but then you lose type checking.

Martin U: Jens solution (changing the code example to have two compare and exchange weak's) has the issue that it requires the representation bytes to stay the same.

Martin S: Using integers would work too.

Niall: intptr_t could not always be considered to be round trip valid.

Philipp: No, the C standard requires the round trip.

Roberto: The pointer could be invalid so what happens with the round trip?

Martin S: For specialized algorithms like this, they could be better done in a library.

Roberto: Is it possible to say there are people in this group that could rewrite this example in a standards compliant manner with the same performance?

Paul: Both Jens (don't agree with the performance equivalence) and Martin S's (loss of diagnostics, but performant) would be problems.

Derek: This is asking for changes that could break optimizations for benefits to a niche market. Not worth it.

David K: There are lock free programming algorithms that have a clear expression but it would unfortunate to have to clutter the code with things like casts to pointer.

Peter: There were many people who worked hard to get this (atomics) in the standards for exactly this reason.

Martin S: The drawback is the inability to detect many cases of end of lifetime of pointers. Going the other way is preventing optimizers from doing optimizations. Perhaps look at one of the other options. Ex. Annotating the code before the free call.

Peter: For option 2 I totally agree with you.

Niall: I am in favor option 2. Any time a compiler can see end of lifetime, it should do anything other than zap the end of lifetime. It would be a good thing if this algorithm would not compile and would really strongly advocate this for any ambiguity.

David K: I would be opposed to that since we are trying to modernize C for multi-core systems. Even if it means giving up diagnostics.

Martin U: Ambiguity is very hard to detect since the problem could happen much later. Ex. Setting a pointer to NULL at the end of the lifetime then much later at a point the compiler could not see it has the failure.

Paul: The advantage of static analyzers is that they have more resource and time to do things than compilers can.

Aaron: It is hard to express what is allocated since it could be hidden to the compiler. Ex. Hidden malloc/free like the Linux sbreak.

Jens: I still agree with option 1 (Derek: Ditto). I also disagree with atomic compare and exchange vs the atomic load being much worse. There is more areas of throttling and other things.

Roberto: I think the idea in the standard that a pointer only makes sense in reference to a storage instance is deeply rooted in the standard. This is why pointer end of lifetime zap make sense. Going via integers makes it explicit that you are looking for the pointer address not the object.

Paul: I disagree this is a good thing for changing existing code.

Aaron: There are cases where malloc(0) returns a non-null pointer and has no allocated storage and it is considered useful.

*Straw poll: Which possible resolution does the committee want among the ones proposed in N2369?

1: 13
2: 11
3: 13
4: 8
5: 0

Paul: I need to come back to prove 2 or 3 is better than the other options.

6.38 Krause, No internal state for mblen [N 2358]

Martin U: If it resets the internal state and it is shared, it would break anyone that did share the state.

Jens: Current standard wording does each function has its own state. Not that it is shared.

Clive: Is the intention to be more like C++?

Philipp: My intention is to make mblen more usable for multithreaded functions.

Aaron: Has anyone talked to C++?

No one in the room has said this.

Philipp: I have been in discussion with the Unicode study group and they like it.

Aaron: I am good with it.

*Straw poll: Do we want N2358 in C2X?

(14/0/0)

Consensus.

6.39 Uecker, Improved Rules for Tag Compatibility [N 2366]

Revision of N2105.

Jens: For the first change to make similar declarations of tagged structures in different scopes compatible would still be different types, even if compatible.

Second part: Allow redefinition in the same scope.

Derek: Did you ever go further as per the reflector discussion with this as templates?

Martin U: No. If someone wants to, I am good with that.

Martin S: Are there any cases where previous valid cases would become invalid?

Martin U: Yes, some edge cases like _Generic would break if they relied on two different structs being incompatible.

Philipp: It could be used to prevent people from accidentally using the wrong struct.

Martin U: No, they'd need to have the same tag.

Philipp: It could still happen if they intended it to be a different type.

Martin U: I consider that unlikely. For non-tagged ones I would like them to stay incompatible.

Derek: The requirement right now is to not require link time checking. Will we need to have linkers to change now?

Martin U: Don't need to. This only adds potential for link time optimizations.

Martin S: I would like to see an example of any problems being discussed.

Martin U: All I'm looking for is do we want to do something like this or not.

Jens: We have the same problem with enumeration types.

Martin U: At least you would solve it for structs.

Example on whiteboard:

TU1: struct s;

TU2: struct s { int x; }

TU3: struct s {double x; }

Derek: If they had a pointer in each declaration then that is interesting.

Martin U: It has a problem since TU3 and TU2 would have different types and referencing either member would be a problem.

Rajan: Any change to tagless structs?

Martin U: No, not yet.

Niall: This is incompatible with C++ right now, but in the future you could add in a concepts line later to make it compatible for cut'n'paste code. It also violates the one definition rule. Half is feasible but half is not. It would make writing C easier.

Aaron: The type compatibility makes sense, but the fact that you can't copy and paste in the same TU vs different TU's is a different mental model.

Niall: Is there room for them to not be completely identical? How do people feel about that?

Martin U: We have non-identical in things like arrays with a size and ones without to generate compatible types. It may happen inside structs too I suppose.

Niall: I just want to have a=b work.

Martin U: Thinks I got from this: Is the incomplete types part necessary or not? Questions about C++ compatibility.

Niall: What this is is defaulted trivial copy constructors.

Martin S: Any thoughts about prototyping this in GCC or another compiler?

Martin U: If the committee wants to go forward I would try. I thought in the mailing list someone said they would be willing to implement this in Clang.

*Straw poll: Does the committee want to see a future paper along the lines of N2366?

(10/0/9)

Consensus: We want to see more papers like N2366.

7. Clarification Requests

One half hour out of each afternoon (to allow for homework)

7.1 Discussion on the Clarification Request Process

Blaine: I don't think given where we are in the process to have a formalized clarification request document given the corrigenda rules. We have a process so people can write a paper. We also now have editor guidelines and can write in LaTeX directly. We don't need or want CR's. The CFP docs likely have no extra value other than a consolidated list.

Martin: I feel we have a process we are familiar with and works for smaller things. For corrections, this process works pretty well.

Blaine: My view is we have a process: Propose it for C2X.

David: We have drained the pipeline enough that the ones left have people writing papers anyways. After C2X goes out we can do this process again.

Martin: I don't think this is the case. Mistakes creep in, and this works for smaller things.

Jens: For the editors, it was not meant to replace the defect report. It was meant for things like the C FP TS integration. I see three things: Things needing tight linking with the editors for new features for example; questions about clarification; editorial problems with the standard. We do need the defect reports for the second bucket.

Blaine: The function of the existing process is to keep track of nits. We kept track of small things. We also had things where we changed our minds. The people who came up with the defect need to push it through the regular process.

Robert S: If we have a radical change to the process, we should manage this as a software development process to track things.

Jens: The Austin group does that.

Aaron: DR is helpful since it says the standard always meant to say this vs a new feature where it is a new language standard mode. It is also useful for our users.

Blaine: The compendium is a place where we could record why we did something or didn't do something.

Rajan: The process we normally use does seem to work for what used to be DR's.

Aaron: Does the author or originator have to be present?

Blaine: No, we have done it in the past where someone else read it. Having a template for Clarification Requests could be flowing through the process normally and not get lost or need champions.

Blaine: I don't really want to have to deal with getting every comma and thing right in the technical corrigenda.

Aaron: Is there some automated way to find out if something is a clarification request?

Blaine: I want to change the process so the defect log will now be clarification requests that come in normally as N papers, but don't need champions. It may come to be a change is needed to the standard, but that will need a new paper with proposed changes to the standard.

Blaine: David used to maintain various standing documents. These clarification requests are basically standing documents that only get appended to.

David: Since C# documents don't change and have version numbers we can use this.

Aaron: Since we reserved C0 as the working draft, can we reserve C1 as the clarification reports.

7.2 IS 9899:2011/9899:2018 Clarification Requests [N 2316]

Blaine: We intend to continue this a document for tracking clarification requests. Fred's format (has the most DR's) is perfect for Clarification requests too. State there may be a problem, give an example, propose a fix. I don't like the general tweaks to words and 6 months to review and then future tweaks, etc. We should just take what's there from the author.

Blaine: This document could track the draft document # of the C standard. Ex. DR476.

Jens: This was probably integrated into the post Pittsburgh draft. All DR's are integrated now.

*AI: Blaine: Ensure all DR's marked C2X are in the latest working draft (N2346).

DR501: Making DECIMAL_DIG obsolescent

*Move DR501 to C2X.

DR496: offsetof questions

N2350 is the proposal with the wording that expresses the intent.

Martin S: C doesn't have conditional diagnostics so I am not happy with this but I will accept it.

Blaine: I don't like that this loses the structure member->subobject which deals with recursion that is lost here in the new paper.

Fred: We need both.

Martin S: Yes, this adds to the N2316 text.

Blaine: I say keep N2316 and deal with N2350 separately.

David K: We should accept the wording in DR496, but if the editors find a better way of saying it they can.

*Move DR496 to C2X.

7.3 TS 18661 Clarification Requests [N 2317]

In addition to normal CR processing, the following items have new material to consider.

The C FP group will now maintain the CR tracking document for this TS after this meeting.

DR13: Move to C2X

DR16: Move to C2X

DR20-25: Move to C2X

Jens: I would like the C FP group to give the changes to me.

Blaine: I would like to see which draft has them folded in like the C2X CR log.

Blaine: I will publish one more version of this and the C2X CR log at the end of this meeting and then end them.

Thomas, P4 CR for rootn case differs from IEEE 754 [N 2309]

Assigned to DR26.

Clive: What happens to the n=0 case?

Fred: It is already there in the other bullets in the document.

*Straw poll: Should N2309 be put into TS 18661-4 and C2X?

(10/0/9)

Consensus to do this. C FP to give the words along with the Part 4a integration words to the editors.

Martin S: Why count abstentions?

David K: It matters when determining consensus.

8. Other Business

8.1 Special math functions (ISO/IEC 24747)

David K: This is the second 5-year review for this document. Typically we stabilize standards at this point. One defect has come up. Similar to the C defect with the date in the version macro. This document has 200808 without the suffix, with the footnote saying it is supposed to be a long int. Originally the committee decided to leave it

since it is still usable.

Martin S: It is nitpicky, this is not saying what it expands to, it is just the value of the macro.

David K: The normative text is fine, it is only the footnote that makes it wrong. I prefer to leave it alone as the editor for this document.

Fred: Do we need to change the reference to C99?

*Straw poll: __STDC_MATH_SPEC_FUNCS__ macro is useful as is. Does the committee want to keep it as is?

(15/0/0)

Consensus to keep it as is.